CHAPTER 1

Inductionless Induction

Hubert Comon

SECOND READERS: Bernhard Gramlich and David Mac Allester and Donald Sanella.

Contents

HANDBOOK OF AUTOMATED REASONING
Edited by Alan Robinson and Andrei Voronkov

1. Introduction

1.1. *Inductive theorem proving*

The aim of inductive theorem proving is to prove statements about particular structures, e.g. natural numbers, lists, arrays... Natural first order axiomatizations of such familiar objects fail to capture the true statements we want to prove.

1.1. EXAMPLE. Assume that we give the following simple definition of $+$ on the natural numbers (in base 1):

$$E = \begin{cases} 0 + x & = & x \\ s(x) + y & = & s(x + y) \end{cases}$$

$x + 0 = x$ is not a logical consequence of the above definition of $+$, as there are models of E which do not satisfy the equation$x + 0 = x$. For instance, consider the interpretation I whose domain is the two elements set $\{0, a\}$ and such that s_I, the interpretation of s in I, is the identity $s_I(0) = 0$, $s_I(a) = a$ and $+_I$, the interpretation of $+$ in I, is the second projection: $u +_I v$ is v. Then the equations of E are satisfied in I. However $x + 0 = x$ is not valid in I, since $a + 0$ equals 0 in I.

The reason of the failure is that our domain of interpretation contains some element (namely a) which cannot be reached from 0 using a number of successors. When writing our specification, our intention was to define the addition on *natural numbers* i.e. terms which are built from 0 using a number of successors. Such a structure will be called *the structure freely generated by* $\{0, s\}$, or (more generally) the *minimal Herbrand model* of the axioms.

If we use the knowledge that we only consider this particular structure, it is not difficult to prove $x + 0 = x$. For instance, we may use our familiar Peano's induction scheme: $0 + 0 = 0$ by the first equation and, if $s^n(0) + 0 = s^n(0)$, then $s^{n+1}(0) + 0 = s(s^n(0) + 0)$ by the second equation and $s(s^n(0) + 0) = s^{n+1}(0)$ by induction hypothesis. Note however that Peano's induction scheme is most naturally expressed in second-order logic.

More generally, most structures of interest in computer science can be defined using first-order terms. For instance, natural numbers can be modeled by $s(s(\ldots s(0)))$ and lists can be modeled by $cons(a_1, cons(a_2 .... cons(a_n, nil)))$. The objects in the structure are represented as terms built over a given signature (the *constructors*), or, more generally, as classes of terms modulo some congruence relation.

1.2. EXAMPLE. Instead of natural numbers as before, we consider a specification of the group of integers; they are represented using successor as before and the negation sign, as usual:

$$(1) \qquad \begin{cases} -0 &= 0 \\ --x &= x \\ s(-s(x)) &= -x \\ 0+x &= x \\ s(x)+y &= s(x+y) \end{cases}$$

The main difference now is that the three first equations specify some relations between the negation sign, 0 and s. Hence the intended structure is no longer freely generated: it is the quotient of the set of terms built on $0, -, s, +$ by the congruence generated by the above set of equations: this is again the *minimal Herbrand model* of the set of axioms. If our specification is "complete" with respect to what we intended, then a set of representatives will consist of $\{s^n(0) \mid n \in \mathbb{N}\} \cup \{-s^n(0) \mid n \in \mathbb{N}\}$. The commutativity $x + y = y + x$ is an inductive consequence of (1). But it is not completely straightforward to see how to prove it.

The question is: how can we (automatically) prove some properties in such minimal Herbrand structures ? One possible approach is to use *induction* as we did in our first example above. Mechanizing induction is precisely the purpose of the chapter XXV. The purpose of the present chapter is to explain a method which achieves the same goal, but without the use of an induction scheme. Let us first review and compare the different methods.

1.2. A few words explaining the title

The expression "inductionless induction" was coined by D. Lankford [Lankford 1981]. It refers to a proof technique in minimal Herbrand models which does not make use of explicit induction rules (hence differs from the inductive proof methods described in chapter XXV): we will stay within classical first-order logic. Inductionless induction should rather be called *proof by consistency*, following D. Kapur and D. Musser's terminology [Kapur and Musser 1987], as we will see. This method was discovered in the years 80-82 by several authors [Musser 1980, Goguen 1980, Lankford 1981, Huet and Hullot 1982] and a lot of work has been devoted to it since, mainly in the term rewriting community. This chapter relies on this work. It aims at giving a general presentation, encompassing most of the results in the field.

The inductionless induction approach is the basis of several inductive theorem provers. In general such theorem provers are small: nothing comparable with the Boyer and Moore theorem prover [Boyer and Moore 1979] in terms of applications. Probably the most well-known one is part of the RRL system [Kapur and Zhang 1995] (this system also includes a cover set induction tool which is not a proof by consistency technique, see below). There are some other implementations such as ReDuX[1] or UNICOM [Gramlich 1990c, Gramlich and Lindner 1991]. However, the main advantage of inductionless induction is the ability to use general purpose first-order theorem provers for inductive theorem proving; there is no real need to develop a dedicated tool. The only requirement is to design dedicated strategies within a first-order theorem prover.

The core of the method for the purely equational case is explained in detail in L. Bachmair's book [Bachmair 1991]. The non-equational case is studied in [Comon and Nieuwenhuis 1998].

Basically, inductionless induction is a "proof by consistency" method. Assume that E is a set of axioms (e.g. the two equations defining $+$ in example 1.1) and that there is (up to isomorphism) a single minimal Herbrand model $\mathcal{H}$ of E (e.g. $\mathbb{N}$, in example 1.1). The idea is to use a (first-order) axiomatization $\mathcal{A}$ of $\mathcal{H}$ such that $\phi \cup \mathcal{A} \cup E$ is consistent if and only if ϕ is valid in $\mathcal{H}$. In example 1.1, such an axiomatization $\mathcal{A}$ can be $\{\forall x.0 \neq s(x);\ \forall x, y.s(x) = s(y) \Rightarrow x = y\}$, as we will see. If we want to prove the conjecture $x + y = y + x$. We add it to $E \cup \mathcal{A}$ and try to derive an inconsistency.

Then, we rely on *saturation techniques* which are introduced in chapter XVII and will be recalled here. In the example, it is possible to saturate $E \cup \mathcal{A} \cup \phi$ without deriving an inconsistency: this proves that commutativity is indeed valid in $\mathbb{N}$. This example is fully developed in example 4.13.

1.3. Inductionless induction versus implicit induction

We chose to restrict ourselves to proofs by consistency, i.e. we do not cover methods using an induction scheme. There are however several techniques which use the ideas developed in the area of proofs by consistency and which also aim at proving inductive theorems. Let us cite for instance the cover set method [Zhang 1988], the test set method [Bouhoula and Rusinowitch 1995] and the rewriting induction method [Reddy 1990]. All these methods are roughly *implicit induction* methods: the induction scheme is not necessarily known beforehand. More precisely, each of these methods provides a set of terms (or a set of pairs (context, term)) which is used for the replacement of the induction variable (depending on the context). Such replacements produce new conjectures which can be simplified by strictly smaller instances of the original conjecture (the induction hypothesis). The proof is completed when all newly generated conjectures are simplified into known inductive theorems.

[1]see `http://www-sr.informatik.uni-tuebingen.de/~buendgen/redux.html`

1.3. EXAMPLE. Consider the following (other) definition of the addition on integers:

$$\begin{cases} 0+x & = & x \\ s(x)+y & = & s(x+y) \\ s(p(x)) & = & x \\ p(s(x)) & = & x \\ p(x)+y & = & p(x+y) \end{cases}$$

All these equations can be used from left to right, yielding a terminating system. Consider the conjecture $x+0=x$. The cover set method will overlap the conjecture with the left members, which yields to three more conjectures: $0+0=0, s(x)+0=s(x), p(x)+0=p(x)$ (in other words we replaced x with all elements of the *cover set* $\{0, s(x), p(x)\}$). The new conjectures can be simplified with the equations defining $+$ and with the induction hypothesis $x+0=x$ in which x is frozen (cannot be instanciated). In our example, $0+0$ reduces to 0 using $0+x=x$, $s(x)+0$ reduces to $s(x+0)$ using the second equation, then to $s(x)$ using the induction hypothesis and finally $p(x+0)$ reduces to $p(x)$ using the last equation and the induction hypothesis. This completes the proof.

The main difference between explicit induction techniques and implicit induction techniques is roughly that the former rely on a semantic ordering while the latter rely on a syntactic ordering (the one which shows the termination of the definitions). However, both use an induction scheme, which is not the case in our chapter.

One should avoid any confusion between such implicit induction techniques and the inductionless induction method. The methods are very different in nature. For instance, in principle, we can prove properties such as $x \neq s(x)$ using inductionless induction. This is not possible with implicit induction methods. The reason is that inductionless induction aims at proving properties in one particular model: the minimal Herbrand model, whereas implicit induction aims at proving properties which hold true in *all* constructor-generated models (e.g. 0 is equal to $s(0)$ in the one-element model of the axioms of example 1.1). The two notions will coincide on positive properties (see lemma 2.2).

Let us insist: one of the main features of inductionless induction is the possibility to use classical first-order theorem provers. Cover set (resp. test set, resp. rewriting) induction are very interesting and useful techniques. However, they use an extra rule, which is not valid in first-order logic and hence, they need dedicated tools. These methods are not relevant to our chapter.

1.4. Outline of the chapter

The subject of this chapter is to explain inductionless induction, to explain how and why it works. More precisions on the setting will be given in section 3 after recalling some basic background in section 2. Then, two main issues have to be discussed:

What are the deduction rules used in this context ? The method originally relies on completion procedures (which are described in chapter XVIII). In the case of proofs by consistency, the deduction system consists in some restrictions of the Knuth-Bendix completion procedure, as explained in [Bachmair 1991, Fribourg 1989].

The Knuth-Bendix completion procedure can be extended to arbitrary clauses with equality, roughly replacing completion with *saturation* and simplification with *redundancy* (see chapters III and XVII). Following this extension, we give in section 4 a general inductive saturation method, which is not restricted to the purely equational case.

How can we build an axiomatization $\mathcal{A}$ which has the desired properties ? Some techniques are presented in section 5. They are mainly reformulations of methods suggested in [Musser 1980, Huet and Hullot 1982, Jouannaud and Kounalis 1986, Bachmair 1988]. Basically, $\mathcal{A}$ expresses that two equal terms should be *ground reducible*. This notion plays such an important role that a whole section (section 6), is devoted to it.

All these techniques have been actually developed in the framework of equational theories. They carry over to clauses with equality, with the proviso that, in general, there is no known axiomatization $\mathcal{A}$ for which the consistency of $e \cup \mathcal{A}$ can be decided for every equation e. Nevertheless, it is possible to design a technique which may fail to prove consistency, but always succeeds in detecting inconsistencies; we get refutationally complete proof techniques (see [Ganzinger and Stuber 1992]). Of course, they may also prove sometimes the consistency of $e \cup \mathcal{A}$. However, in the clausal case, it is not only hard to prove consistency, but also to get a saturated set.

As a conclusion, we develop in section 7 a short (maybe controversial) comparison between the different methods.

2. Formal background

2.1. Terms and clauses

We will use finite *terms* constructed out of a finite set of *function symbols* F whose arity (number of arguments) is fixed and an infinite set of *variables* X . The set of terms is written $T(F, X)$. The set of variables of a term is written $\mathcal{V}ars(t)$. A *ground term* is a term which does not contain any variable. The set of ground terms is written $T(F)$. The identity (uninterpreted equality) on terms is written $\equiv$. The terms are also identified, as usual, with mapping from prefix-closed sets of strings of natural numbers called *positions* to $F \cup X$. The label at a position p of t is written $t(p)$. The subterm of t at position p is denoted $t|_p$.

Substitutions are (simultaneous) replacements of variables with terms. Concerning substitutions, we use the same definitions and notations as in chapter XV. In particular, when $x_1, ..., x_n$ are variables, $\{x_1 \mapsto t_1; ...; x_n \mapsto t_n\}$ maps each x_i to t_i and the other variables to themselves. A *ground substitution* maps every variable to a ground term. A term t *encompasses* a term u if there is an instance of u which is a subterm of t.

An *atomic formula* (also called a *positive literal*) is either an equation $s = t$ between two terms or a predicate symbol applied to terms $P(t_1, \ldots, t_n)$. A *literal* is either an atomic formula or its negation. A (general) *clause* is a disjunction of literals. A *positive clause* is a disjunction of positive literals. A *Horn clause* is a clause which contains exactly one positive literal.

2.2. Equational deduction

If E is a set of equations between terms of $T(F, X)$, $=_E$ is the smallest congruence on $T(F, X)$ such that $s\sigma =_E t\sigma$ for all equations $s = t \in E$ and for all substitutions σ.

Given a set of equations E, $s = t$ is a *consequence* of E if $s =_E t$. Equivalently $s = t$ is a consequence of E if it can be proved using a finite sequence of *replacement of equals by equals*: $=_E$ is the reflexive and transitive closure of the binary relation $\underset{E}{\longleftrightarrow}$. $u \underset{E}{\longleftrightarrow} v$ if there is an equation $l = r \in E$ and a substitution σ such that u is obtained from v by replacing some occurrence of $l\sigma$ with $r\sigma$ (resp. replacing some occurrence of $r\sigma$ with $l\sigma$). If we want to make precise the equation and/or the substitution, we can use them as indices of the relation as in: $u \overset{\sigma}{\underset{l=r}{\longleftrightarrow}} v$.

2.3. Inductive theory

A *Herbrand interpretation* of a set of first-order sentences is an interpretation whose domain is a quotient of the set of ground terms $T(F)$ by some congruence relation and such that the interpretation of every ground term t is the congruence class of t. In such interpretations, every element of the domain can be constructed from elements in the signature F; coming back to example 1.1, the interpretation I is not an Herbrand interpretation since a cannot be constructed from $0, s, +$ only. Inductive theorems are the formulas which can be proved using E and an axiom scheme which rules out interpretations which are not Herbrand interpretations. This leads us to the following definition:

2.1. DEFINITION. Let E be a set of first-order sentences. A first-order formula ϕ is an *inductive consequence* of E iff, for every Herbrand interpretation $\mathcal{H}$, $\mathcal{H} \models E$ implies $\mathcal{H} \models \phi$.

The set of all inductive consequences of E is called the *inductive theory of* E . Inductive consequences of E will also be called *inductive theorems* in what follows.

When E is a set of Horn clauses with equality, there is a (unique) smallest (minimal) Herbrand model of E, which we write $\mathcal{I}_E$. $\mathcal{I}_E$ is the intersection of all Herbrand models of E (hence its minimality): for each predicate symbol P, a tuple of ground terms belongs to the interpretation of P in $\mathcal{I}_E$ iff it is the case in all

Herbrand models of E. $\mathcal{I}_E$ is also the least fixed point of E: consider the immediate consequence operator T_E which maps a Herbrand interpretation $\mathcal{H}$ to a Herbrand interpretation $\mathcal{H}'$ as follows. An atom $P(t_1, \ldots, t_n)$ belongs to $\mathcal{H}'$ iff either it is in $\mathcal{H}$, or it is a consequence of stricly smaller (w.r.t. size) equalities in $\mathcal{H}'$ or else there is a clause $B \Rightarrow P(u_1, \ldots, u_n)$ in E and a ground substitution σ such that $\mathcal{H} \models B\sigma$ and $P(t_1, \ldots, t_n) \equiv P(u_1, \ldots, u_n)\sigma$. $\mathcal{I}_E$ is the union of $T_E^n(\emptyset)$ for $n \in \mathbb{N}$. Finally, $\mathcal{I}_E$ is also called the *initial model* of E because it is an initial object in the category of models of E.

When E is a set of Horn clauses, we do not need to consider all the Herbrand interpretations, but only the smallest one[2]:

2.2. LEMMA. *Let E be a set of Horn clauses and c be a positive clause. c is an inductive consequence of E iff $\mathcal{I}_E \models c$.*

When E is simply a set of (implicitly universally quantified) equations, $\mathcal{I}_E$ is the quotient algebra $T(F)/_{=_E}$.

2.3. EXAMPLE. In example 1.1, $x + 0 = x$ is not a consequence of E, but it is an inductive consequence since for every ground term t, $t + 0 =_E t$.

Inductive consequences of E should not be confused with elements in the theory of $\mathcal{I}_E$. For instance, $x \neq s(x)$ is *not* an inductive consequence of the set E of example 1.1 since the trivial one element model of E does not satisfy $x \neq s(x)$. However, $\mathcal{I}_E \models x \neq s(x)$. The inductive theory is contained in the theory of $\mathcal{I}_E$, but the converse is false. In what follows, we will prove or disprove conjectures in the theory of $\mathcal{I}_E$. This coincide with inductive consequences for positive conjectures only. However, we also consider the possibility of non-positive conjectures, in which case the results are also interesting, though having a different meaning. For instance, what is known in logic programming as *negation as failure* will be an instance of our proof by consistency method: if the (negative) conjecture is consistent with E (finite Failure of the derivation of an empty clause), then it is valid in $\mathcal{I}_E$. That is exactly what negation as failure says.

2.4. Constructors and sufficient completeness

The problem we encountered in example 1.2 is the absence of *free constructors*, i.e. the model we have in mind cannot be freely generated by (some) symbols in the alphabet. Let us make this notion more formal.

2.4. DEFINITION. A subset C of F is called a set of *constructors* (w.r.t. E) if, for every term t in $T(F)$, there is a term u in $T(C)$ such that $E \models t = u$.

2.5. EXAMPLE. $\{0, s\}$ is a set of constructors in example 1.1.

[2]Similarly, for arbitrary clauses, we do not need to consider all Herbrand interpretations, but only the minimal ones.

2.6. DEFINITION. E is *sufficiently complete* with respect to a subset D of F if $C = F \setminus D$ is a set of constructors w.r.t. E. Then elements of D are called *defined symbols* .

2.7. EXAMPLE. Assume that we have, in addition to example 1.1, a new function symbol p and the three additional equations:

$$\begin{cases} p(s(x)) & = & x \\ s(p(x)) & = & x \\ p(x) + y & = & p(x+y) \end{cases}$$

Then E is sufficiently complete w.r.t. $\{+\}$.

If we know a set of constructors, then we may use a structural induction over this set of constructors in order to prove inductive consequences of E. However, this is often not sufficient: F itself is always a set of constructors. In example 1.1, if we try to prove $x + 0 = x$ by structural induction on F, it will be hard to succeed because there are a lot of necessary lemmas (such as the associativity of $+$) which are in turn hard to prove using such a structural induction.

One can argue that F is really a bad choice. We could choose a *minimal* set of constructors. However, on one hand such a minimal set is not easy to find and, on the other hand, it is not sufficient in many cases. This can be illustrated by example 1.2: $\{0, s, -\}$ is a minimal set of constructors (how do you prove it ?). Trying to prove $x + 0 = x$ by structural induction on the set of constructors is not so easy, and requires some lemmas (which have to be discovered).

2.8. DEFINITION. A set C of constructors is *free* if $E \cup \{s \neq t \mid s, t \in T(C), s \not\equiv t\}$ is consistent.

When E is a set of Horn clauses, C is a set of constructors iff for every distinct terms $s, t \in T(C), E \not\models s = t$. In example 1.1, the set $C = \{0, s\}$ of constructors is *free* because, for any two distinct terms $s, t \in T(C)$, $s \neq_E t$. However, in examples 1.2 and 2.7, any set of constructors is not free.

2.5. Term Rewriting

This section and the next one are devoted to a detour through term rewriting. Though this detour is not necessary for our purpose, many examples will use this framework. We only recall very briefly the basic background. More information and developments can be found in e.g. [Bachmair 1991, Dershowitz and Jouannaud 1990] and in the chapter XVIII. The reader who is familiar with term rewriting systems can jump to section 3.

A *rewrite rule* is an oriented equation. A *rewrite system* is a set of rewrite rules. Rewriting a term t into a term u using a rule $l \to r$ with the substitution σ is defined as the replacement of equals by equals, except that only $l\sigma$ can be replaced with $r\sigma$

and not the converse. This relation is written $t \xrightarrow[l \to r]{\sigma} u$. A rewrite system $\mathcal{R}$ defines a *rewrite relation* $\xrightarrow[\mathcal{R}]{}$ which is the union of all $\xrightarrow[l \to r]{\sigma}$ for all substitutions σ and rules $l \to r$ in $\mathcal{R}$. The corresponding *reduction relation* is the reflexive transitive closure $\xrightarrow[\mathcal{R}]{*}$ of $\xrightarrow[\mathcal{R}]{}$.

$\mathcal{R}$ is *confluent* if, for all terms s, t, u such that $s \xrightarrow[\mathcal{R}]{*} t$ and $s \xrightarrow[\mathcal{R}]{*} u$, there is a term v such that $t \xrightarrow[\mathcal{R}]{*} v$ and $u \xrightarrow[\mathcal{R}]{*} v$. $\mathcal{R}$ is *terminating* iff there is no infinite sequence $t_1 \xrightarrow[\mathcal{R}]{} \dots \xrightarrow[\mathcal{R}]{} t_n \xrightarrow[\mathcal{R}]{} \dots$. $\mathcal{R}$ is *convergent* when it is both terminating and confluent. When $\mathcal{R}$ is convergent, each term t has a unique normal form $t\downarrow_{\mathcal{R}}$. For convergent rewrite systems $\mathcal{R}$, the equational theory is decidable: $s =_{\mathcal{R}} t$ iff $s\downarrow_{\mathcal{R}} \equiv t\downarrow_{\mathcal{R}}$. (See chapter XVIII for more details).

2.6. Standard Completion

Completion processes (which are explained in more detail in chapter XVIII) consist basically in orienting equations according to a given reduction ordering and computing equational consequences. They, roughly, aim at computing from a set of equations an equivalent convergent rewrite system.

More precisely, a completion procedure takes as input a set of equations E and a reduction ordering $\succeq$ and applies successively one of the rules given in chapter **??**:

Deduce which adds some critical pairs

Delete,Collapse,Compose,Simplify which simplify the rules and the equations

Orient which turns equations into rules according to the ordering.

A completion procedure is fair whenever some possible deduction or orientation is not indefinitely delayed. A completion procedure may either *fail* because some unorientable and unsimplifiable equation is found (in which case the completion cannot be fair). Otherwise, the ultimate set of persisting rewrite rules R_∞ is a convergent rewrite system (see [Dershowitz 1990, Bachmair 1991]).

Examples of completion can be found in chapter XVIII of this book. Let us give here a simple example:

2.9. EXAMPLE. We consider the presentation of integers in example 1.2. Using a recursive path ordering extending the precedence $+ > - > s > 0$, every equation can be turned into a rewrite rule and we get the rewrite system:

$$\mathcal{R} = \left\{ \begin{array}{rcll} -0 & \to & 0 & (1) \\ --x & \to & x & (2) \\ s(-s(x)) & \to & -x & (3) \\ 0+x & \to & x & (4) \\ s(x)+y & \to & s(x+y) & (5) \end{array} \right.$$

Overlapping (3) with itself we get

$$s(x) \underset{(2)}{\longleftarrow} s(--x) \underset{(3}{\longleftarrow} s(-s(-s(x))) \underset{(3)}{\longrightarrow} --s(x) \underset{(2)}{\longrightarrow} s(x)$$

yielding a critical pair which is simplified and deleted. Overlapping (3) and (5):

$$s(-s(x)+y) \underset{(5)}{\longleftarrow} s(-s(x))+y \underset{(3)}{\longrightarrow} -x+y$$

we get the critical pair $s(-s(x)+y) = -x+y$ which can be turned into a new rule:

$$s(-s(x)+y) \rightarrow -x+y \qquad (6)$$

There are new critical pairs: (3) + (6) gives

$$s(x{+}y) \underset{5}{\leftarrow} s(x){+}y \underset{(2)}{\longleftarrow} --s(x){+}y \underset{(6)}{\longleftarrow} s(-s(-s(x)){+}y) \underset{(3)}{\longrightarrow} s(--x{+}y) \underset{(2)}{\longrightarrow} s(x{+}y)$$

which is simplified and deleted. (6) +(3) gives

$$-(-s(x)+y) \underset{(3)}{\longleftarrow} s(-s(-s(x)+y)) \underset{(6)}{\longrightarrow} s(-(-x+y))$$

which is turned into a new rule

$$-(-s(x)+y) \rightarrow s(-(-x+y)) \qquad (7)$$

(7) +(2) yields

$$-s(-(-(x+y))) \underset{(7)}{\longleftarrow} --(-s(x)+y) \underset{(2)}{\longrightarrow} -s(x)+y$$

which is turned into the new rule

$$-s(x)+y \;\rightarrow\; -s(-(-x+y)) \qquad (8)$$

(6) is simplified using (8):

$$s(-s(x)+y) \underset{(8)}{\longrightarrow} s(-s(-(-x+y))) \underset{(3)}{\longrightarrow} --(-x+y) \underset{(2)}{\longrightarrow} -x+y$$

and then deleted. Similarly, (8) simplifies (7) which is deleted and all remaining critical pairs can be simplified and deleted. We get the convergent rewrite system:

$$\left\{\begin{array}{rcl} -0 & \rightarrow & 0 \\ --x & \rightarrow & x \\ s(-s(x)) & \rightarrow & -x \\ 0+x & \rightarrow & x \\ s(x)+y & \rightarrow & s(x+y) \\ -s(x)+y & \rightarrow & -s(-(-x+y)) \end{array}\right.$$

3. General Setting of the Inductionless Induction Method

For simplicity, we assume from now on, in the rest of this chapter, that E is a finite set of Horn clauses. To some extent, it is possible to generalize the approach to arbitrary clauses, considering a *perfect model* semantics . The interested reader is referred to [Comon and Nieuwenhuis 1998].

The method roughly works as follows: given a set of clauses (with equality) E, and a set of conjectures $\mathcal{C}$, we add $\mathcal{C}$ to E and try to derive an inconsistency. If this turns out to be impossible, then $\mathcal{C}$ is an inductive consequence of E. "Inconsistency" here has to be understood w.r.t. some axiomatization $\mathcal{A}$ of $\mathcal{I}_E$.

3.1. EXAMPLE. Let us consider the simplest example (example 1.1): Let $\mathcal{A}$ be the set of (universally quantified) formulas:

$$\left\{ \begin{array}{ccc} & s(x) \neq 0 & \\ s(x) = s(y) & \Rightarrow & x = y \end{array} \right.$$

$\mathcal{I}_E \models \mathcal{A}$. Now, assume that we conjecture $s(x) + 0 = s(0)$. Adding this equation to E, we can derive

$$s(0) \xleftrightarrow[s(x)+0=s(0)]{} s(s(y)) + 0 \xleftrightarrow[E]{} s(s(y) + 0) \xleftrightarrow[E]{} s(s(y + 0)).$$

i.e. for all y, $s(0) = s(s(y + 0))$. Then using the second formula of $\mathcal{A}$, we get $(\forall y).s(y+0) = 0$, which contradicts the first formula of $\mathcal{A}$. This shows that $s(x)+0 = s(0)$ is not an inductive consequence of E.

More generally, if $\mathcal{A}$ is a set of formulas such that $\mathcal{I}_E \models \mathcal{A}$ and if $\mathcal{A} \cup E \cup \mathcal{C}$ is inconsistent, then $\mathcal{C}$ cannot be a set of inductive consequences of E. We also want some converse implication, allowing to derive the inductive validity of $\mathcal{C}$ from the consistency of $\mathcal{A} \cup E \cup \mathcal{C}$. This is summarized in the following properties of $\mathcal{A}$:

3.2. DEFINITION. An *I-axiomatization* of $\mathcal{I}_E$ is a recursive set $\mathcal{A}$ of purely universal formulas such that:

1. $\mathcal{I}_E \models \mathcal{A}$
2. For every Herbrand model $\mathcal{M}$ of E

$$\mathcal{M} \models \mathcal{A} \text{ implies } \mathcal{M} \text{ is isomorphic to } \mathcal{I}_E$$

3.3. EXAMPLE. The set $\mathcal{A}$ of example 3.1 is an I-axiomatization of the corresponding algebra $\mathcal{I}_E$.

Of course such axiomatizations are, in general, incomplete since the first-order theory of $\mathcal{I}_E$ may be not recursive. However, they have a nice property which is the basis of the method:

3.4. LEMMA. *Let $\mathcal{A}$ be an I-axiomatization of $\mathcal{I}_E$. If $E \cup \mathcal{C} \cup \mathcal{A}$ is consistent then $\mathcal{I}_E \models \mathcal{C}$. Hence, if $\mathcal{C}$ is a set of positive clauses, it is an inductive consequence of E.*

PROOF. If $E \cup \mathcal{C} \cup \mathcal{A}$ is consistent, then it has at least one Herbrand model because it is a set of purely universal sentences. Now, by property 2 of definition 3.2, $E \cup \mathcal{C} \cup \mathcal{A}$ has at most one Herbrand model, up to isomorphism. Hence $E \cup \mathcal{C} \cup \mathcal{A}$ has exactly one Herbrand model, which is $\mathcal{I}_E$. $\square$

This is a key lemma since, roughly, it reduces (via $\mathcal{A}$) inductive consequences to first-order consistency.

It remains, however, to clarify several problems. First, how is it possible to check consistency (or inconsistency) of $E \cup \mathcal{C} \cup \mathcal{A}$? Then how is it possible to find I-axiomatizations?

We delay the problem of finding I-axiomatizations until the next sections (section 5 is dedicated to a number of examples for $\mathcal{A}$). With respect to the first issue, a first remark is that we do not need to consider inferences whose premisses are conjectures only (we may use a *linear strategy*). The reason is given by the following simple lemma:

3.5. LEMMA. *Let $\mathcal{A}$ be an I-axiomatization. Then $\mathcal{A} \cup E \cup \mathcal{C}$ is inconsistent iff there is a clause $c \in \mathcal{C}$ and a ground substitution σ such that $c\sigma \cup \mathcal{A} \cup E$ is inconsistent.*

PROOF. Simply, if $c\sigma \cup \mathcal{A} \cup E$ is consistent, then $\mathcal{I}_E$ is a model of these formulas. $\square$

At this point, we may use any refutation-complete first-order deduction system, together with a linear strategy. If $\mathcal{C}$ is not an inductive consequence of E, then an inconsistency proof will be found. However, if $\mathcal{C}$ is an inductive consequence of E, we should have a chance to complete the proof. In other words, we need to be able to prove the consistency. As already explained, the idea will be to saturate the set of clauses: a saturated set of clauses is consistent if and only if it does not contain the empty clause. Unfortunately, the saturation process often runs forever. Hence any restriction on the consequences which have to be computed is welcome. One example is the above linear strategy. There is also one property which is known and not yet exploited: we know that $\mathcal{A} \cup E$ is consistent. Moreover, roughly, $\mathcal{A}$ contains only "negative" information (stating what should be distinct in the model). We may exploit these informations. For instance, we will perform the (in)consistency proofs in two stages: first deductions on $\mathcal{C} \cup E$, yielding a possible inconsistency witness c, then the (in)consistency proof of $c \cup \mathcal{A}$.

Such additional restrictions on the consequences which have to be computed are captured in the inductive saturation strategies which we are going to introduce now.

4. Inductive completion methods

This section is devoted to the deduction engine in the framework of inductive proofs by consistency methods. Historically, the deduction engine was the Knuth-Bendix completion procedure: E was a set of equations, as well as $\mathcal{C}$. Therefore, all prototype implementations and published examples of this proof by consistency approach rely on the completion of term rewriting systems. We will however try to give a more general view of the deduction systems, without restricting ourselves to the equational case. As we have seen, inductive proofs can be reduced to some particular (in)consistency proofs which can be performed by any first-order theorem prover. The only problem is that, in general, such provers are designed for *refutation*, i.e. *inconsistency* proofs. We need here a first-order theorem prover which is (also) able to conclude the *consistency* of a set of formulas. Saturation based theorem provers are adequate for this purpose. The goal now is to design some dedicated strategies which can be used with a saturation-based theorem prover in the context of inductive proofs.

First, we recall the notions of redundancy and saturation in section 4.1. Next, we state a general framework of inductive completion (or inductive saturation) in section 4.2. Then we propose a set of deduction rules for inductive completion in section 4.3. What we are presenting covers the inductive completion methods described in [Bachmair 1991]; when E and $\mathcal{C}$ are equations we find these completion techniques as particular instances of what is presented here. In section 4.4, we show that the deduction rules of section 4.3 satisfy the desired properties, provided that $\mathcal{A}$ is a so-called "normal axiomatization" (a particular class of I-axiomatization which will be defined there). Section 4.5 is devoted to a number of examples, while section 4.6 investigates some further refinements of the strategy.

4.1. Redundancy and Saturation

The notions of redundancy and saturation are studied in detail in chapter XVII (see **??**). Roughly, a clause c is redundant in a set of clauses $\mathcal{S}$ if all its ground instances are consequences of strictly smaller clauses in $\mathcal{S}$. A set of clauses is saturated (w.r.t. an inference system) if all inferences yield redundant clauses. We strengthen these notions in our framework of proofs by consistency: we will say that a clause c is redundant in $\mathcal{S}$ if all its ground instances are consequences of strictly smaller clauses in $\mathcal{S}$ *and* any instances of inductively valid clauses. This is important since we have more redundant clauses, hence it is easier to get a saturated set.

We assume that $\succeq$ is a reduction ordering that is total on ground terms (i.e. an ordering which is well-founded, total, and monotonic: $t \succ u$ implies $f(\dots t \dots) \succ f(\dots u \dots)$ for every function symbol f; see chapter XVIII for more on reduction orderings). This ordering is extended into a total ordering on ground clauses, such that ground equalities are smaller than any other ground literal. . Given a ground clause C, $C^{\prec}$ is the set of ground clauses that are strictly smaller than C in this ordering.

4.1. EXAMPLE. Assume that we have the function symbols $0, s, +$ and one binary predicate symbol gt (besides equality). The ground terms can be ordered according to the lexicographic path ordering extending the precedence $+ > s > 0$: $0 < s(0) \ldots < s^n(0)$, any term containing $+$ is larger than a term not containing $+$ and $s_1 + s_2 < t_1 + t_2$ iff either $s_1 < t_1$ and $s_2 < t_1 + t_2$ or $s_1 = t_1$ and $s_2 < t_2$ or else $s_1 + s_2 \leq t_2$. Then

$$s(0) + 0 = 0 < gt(0,0) < gt(0 + (s(0) + s(0)), s(0)) < gt(s(0) + 0, 0)$$

The clause $s(0 + x) = s(x)$ is redundant in $\{0 + x = x\}$.

4.2. DEFINITION. A ground conjecture c is *redundant* in a set of conjectures $\mathcal{C}$ if $E \cup \mathcal{A} \cup \mathcal{C}^{\prec c} \models c$. A non-ground conjecture is redundant if all its ground instances are redundant.

An inference is redundant in $\mathcal{C}$ if one of its premisses or its conclusion is redundant in $\mathcal{C}$.

Note here that, for redundancy, we may use the axioms E and $\mathcal{A}$ without any restriction, which makes a difference with the notions defined in chapter XVII.

This notion can be refined, using well-known techniques in the field of saturation-based theorem proving [Bachmair and Ganzinger 1994, Nieuwenhuis and Rubio 1995]. For instance, instead of comparing the instances of the clauses, we may compare the pairs (c, σ), which allows to capture subsumption.

4.3. EXAMPLE. Here is an ordering which, specialized to the pure equational case, gives the proof ordering of [Bachmair 1988].

We define an ordering $\gg$ on pairs (c, σ) where c is a clause and σ is a substitution as follows: $(c, \sigma) \gg (c', \sigma')$ iff the multiset of pairs (L, σ) for atomic formulas $L \in c$ is larger than the multiset of pairs (L', σ') for atomic formulas $L' \in c'$. The multisets are compared according to the multiset extension of the following ordering:

- For every predicate symbol P distinct from the equality,

$$(P(t_1, \ldots, t_n), \sigma) \gg (s = t, \sigma')$$

- for every predicate symbols P, Q distinct from the equality, $(P(t_1, \ldots, t_n), \sigma) \gg (Q(s_1, \ldots, s_m), \sigma')$ iff
  either $P(t_1, \ldots, t_n)\sigma \succ Q(s_1, \ldots, s_m)\sigma'$
  or else $P(t_1, \ldots, t_n)\sigma = Q(s_1, \ldots, s_m)\sigma'$ and $P(t_1, \ldots, t_n) \succ Q(s_1, \ldots, s_m)$.
- $(s = t, \sigma) \gg (u = v, \sigma')$ iff $K(s = t, \sigma) \gg K(u = v, \sigma')$ where K is defined by:

$$\begin{cases} K(s = t, \sigma) &= (\{s\sigma\}, s, t\sigma) & \text{if } s \succ t \\ K(s = t, \sigma) &= (\{t\sigma\}, t, s\sigma) & \text{if } t \succ s \\ K(s = t, \sigma) &= (\{s\sigma, t\sigma\}, s = t, \bot) & \text{otherwise} \end{cases}$$

and triples are ordered using the lexicographic extension of the multiset extension of $\succeq$, the encompassment ordering, $\succeq$ respectively. $\perp$ is the smallest ground term.
Using such a refinement, a clause $c\sigma$ is redundant in a set of clauses which contains c because, for every literal L and every ground substitution θ, $(L\sigma, \theta) \gg (L, \sigma\theta)$.
Similarly, if an equation $s = t$ is reducible by a rule $l \rightarrow r$ (with $lsuccr$), then $s = t$ is redundant in any set of clauses containing $l = r$ and the simplified version $s' = t'$ of $s = t$. Hence we may *simplify* $s = t$, replacing it with $s' = t'$, as it is usual in term rewriting.

Finally, a set of clauses is *saturated* (with respect to a given redundancy criterion and an inference system Inf) if all inferences are redundant. In what follows, by "saturated" we will implicitly assume the inference system $\mathcal{I}$ of chapter XVII.

With a set of clauses E is associated a convergent rewrite system $\mathcal{R}_E$ on ground terms defined by $s \rightarrow t \in \mathcal{R}_E$ iff $E \models s = t$ and t is minimal (w.r.t. $\succeq$) among the terms $\{u \mid u \not\equiv s,\ E \models s = u\}$. For saturated sets of clauses, $\mathcal{R}_E$ is the interreduced version of the definition given in **??**.

From results on saturated sets of clauses, we get immediately the following:

4.4. LEMMA. *If E is saturated and $E \models u = v$ for two ground terms u, v, then either $u \equiv v$ or else there is a ground instance $D \vee l = r$ of a clause in E such that $l > D, r$, $E \not\models D$ and $u = v$ is reducible by $l \rightarrow r$.*

4.2. Inductive saturation strategies

The first definition expresses the ability to split the (in)consistency proofs into two parts: deductions between $\mathcal{C}$ and E on one hand and (in)consistency proofs of $\mathcal{C} \cup \mathcal{A}$ on the other hand.

4.5. DEFINITION. Let $\mathcal{A}$ be an I-axiomatisation of $\mathcal{I}_E$. An *inductive saturation strategy* is a mapping $\mathcal{S}$ which associates each pair $E, \mathcal{C}$ with a set of clauses $\mathcal{U}$ such that

1. Each element of $\mathcal{U}$ is a logical consequence of $E, \mathcal{C}$
2. If $\mathcal{S}(E, \mathcal{C}) = \emptyset$, then $\mathcal{C}$ is an inductive consequence of E iff for every $c \in \mathcal{C}$, c is consistent with $\mathcal{A}$.

For instance, if $\mathcal{S}$ computes all non-redundant consequences of $E, \mathcal{C}$, then $\mathcal{S}(E, \mathcal{C}) = \emptyset$ means that $E \cup \mathcal{C}$ is saturated. More generally, $\mathcal{S}$ may compute some of the non-redundant consequences only, provided that they are sufficient to detect inconsistencies. A typical such example is given by *linear strategies* : if we start from a set of axioms which is saturated, only overlaps of the axioms into the conjectures suffice to detect inconsistencies. This is the basis for instance of narrowing strategies (see [Slagle 1974] for an early reference or the chapter XVII). And, indeed, the idea of $\mathcal{S}$

is to narrow the conjectures until we find a counter-example (or no other deduction is possible).

Given an inductive saturation strategy $\mathcal{S}$ a *derivation sequence* is a sequence $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n, \ldots$ such that $\mathcal{C} = \mathcal{C}_0$ and each $\mathcal{C}_{i+1}$ is obtained from $\mathcal{C}_i$ by removing some redundant clause or by adding to $\mathcal{C}_i$ some $c \in \mathcal{S}(E, \mathcal{C}_i)$.[3]

A derivation sequence is *fair* (w.r.t. $\mathcal{S}$) if every clause which can be persistently derived is eventually derived:

$$c \in \bigcap_i \bigcup_{j \geq i} \mathcal{S}(E, \mathcal{C}_j) \Rightarrow \exists j, c \in \mathcal{C}_j.$$

A reformulation of the definitions gives:

4.6. LEMMA. *Every inductive saturation strategy is refutationally complete: if $\mathcal{C}$ is not a set of inductive theorems, then, for every fair derivation sequence, there is an index i and a clause $c \in \mathcal{C}_i$ such that $c, \mathcal{A} \vdash \Box$.*

Now, we have to show some instances of this general scheme: we have to provide some inductive saturation strategies.

4.3. Ordered strategies

We consider a superposition calculus: we use an inference system which is a restriction of ordered paramodulation (as defined in chapter XVII). If there are non-equational atoms $P(\ldots)$, we can encode them as equations $P(\ldots) = true$. We prefer here to avoid the encoding for two reasons: first, examples are more naturally expressed with the predicate representation, second, additional restrictions can be put on the application of inference rules for such literals. Therefore, we are also using restricted forms of ordered resolution.

In our framework, we add a strong restriction on the possible deductions: we use a linear strategy (as for so-called *narrowing*): only deductions overlapping an axiom (from E) into a conjecture are allowed. We give below the inference rules.

Conjecture superposition

$$\frac{D \vee l = r \qquad\qquad C \vee L[s]}{D\sigma \vee C\sigma \vee L[r]\sigma}$$

if
- $\sigma = \mathrm{mgu}(l, s)$,
- s is not a variable,

[3]For those who are familiar with the subject, such a definition rules out the possibility to simplify an axiom using a conjecture. This is for sake of simplicity only. We can of course derive pairs $(E_i, \mathcal{C}_i)$.

- $r\sigma \not\succeq l\sigma$, $D\sigma \not\succeq l\sigma$.
- $D \vee l = r \in E$
- $C \vee L[s] \in \mathcal{C}$

[Equality resolution]

$$\frac{C \vee s \neq t}{C\sigma}$$

if
- $\sigma = \text{mgu}(s, t)$
- $C \vee s \neq t \in \mathcal{C}$

Ordered binary resolution

$$\frac{D \vee P(s_1, \ldots, s_n) \qquad C \vee \neg P(t_1, \ldots, t_n)}{D\sigma \vee C\sigma}$$

if
- $\sigma = \text{mgu}(P(s_1, \ldots, s_n), P(t_1, \ldots, t_n))$,
- $D \vee P(s_1, \ldots, s_n) \in E$,
- $C \vee \neg P(t_1, \ldots, t_n) \in \mathcal{C}$,
- $P(s_1, \ldots, s_n)\sigma \not\leq D\sigma$.

[Ordered factoring]

$$\frac{C \vee L \vee L'}{C\sigma \vee L\sigma}$$

if
- $\sigma = \text{mgu}(L, L')$
- $L\sigma$ is maximal in $(C \vee L \vee L')\sigma$
- $C \vee L \vee L' \in \mathcal{C}$

For all inference rules, the result of the inference is added to $\mathcal{C}$.

Remarks
- Redundancy criteria based on the same ordering can be used as well: inferences are restricted to non-redundant inferences and we may use deletion of redundant conjectures.
- In our framework, the redundancy criteria can be further refined, using in an unrestricted way formulas that are known to be valid in $\mathcal{I}_E$.

- Though superpositions are restricted to conjectures and axioms, we may well use both lemmas which are known to be inductive theorems and axioms in $\mathcal{A}$. This often yields simplifications.
- The above inference rules allow superpositions on the non-maximal sides of the equations in the conjectures. We may impose the superposition on the maximal side of negative literals without changing the results. However, the classical inductive completion methods for equations only considers superpositions on the maximal sides of equations (hence positive literals) [Bachmair 1988]. This restriction can be added to the inference system, with two provisions: we have to add an equality resolution inference rule (and an ordered factorization). Moreover, a slightly stronger property on axiomatizations is needed (see [Comon and Nieuwenhuis 1998] for more details). Then our inference system above is a generalization fo the classical inductive Knuth-Bendix completion.
- Further restrictions can be imposed without altering the results. They are discussed in section 4.6

Actually, we could also imagine other inference rules and other strategies. What is important here is the idea of narrowing: we want to deduce the minimal counter-examples from false conjectures. More precisely, we want to keep the following property:

4.7. LEMMA. *Assume that E is a saturated set of Horn clauses and $\mathcal{C}$ is a set of conjectures. Then*

1. *If $c\sigma$ is a minimal ground instance of a clause in $\mathcal{C}$ such that $c\sigma \cup \mathcal{A} \cup E$ is inconsistent and if $c\sigma$ is reducible by $\mathcal{R}_E$, then there is a non-redundant inference $c, E \vdash c'$ and a ground substitution σ' such that $c'\sigma' \cup \mathcal{A} \cup E$ is inconsistent and $c'\sigma' \prec c\sigma$.*
2. *If $E \models P(s_1, \ldots, s_n)\sigma$ and if $c\sigma \equiv \neg P(s_1, \ldots, s_n)\sigma \vee c_1\sigma$ is a minimal ground instance of a conjecture which is irreducible by $\mathcal{R}_E$ and such that $c\sigma \cup \mathcal{A} \cup E$ is inconsistent, then there is a non-redundant inference $c, E \vdash c'$ and a ground substitution σ' such that $c'\sigma' \cup \mathcal{A} \cup E$ is inconsistent and $c'\sigma' \prec c\sigma$.*

PROOF. (sketch) We only consider the first property: the second one is similar. σ is irreducible by minimality of $c\sigma$. Then $c\sigma$ is reducible at a non-variable position p by a rule $l\theta \to u \in \mathcal{R}_E$. Assume moreover, without loss of generality, that $l\theta$ is irreducible by any other rule. E being saturated, by lemma 4.4, there is a clause $D \vee l = v \in E$ with $l\theta \succ v\theta, D\theta$. Then there is an inference $c, D \vee l = v \vdash c[v]_p\tau \vee D\tau$ by conjecture superposition, yielding a clause c' whose instance $c\sigma[v\theta]_p \vee D\theta$ is strictly smaller than $c\sigma$, and which is also inconsistent with $E \cup \mathcal{A}$. The inference is not redundant, otherwise $c\sigma$ itself would be redundant in $\mathcal{C}$, which would contradict its minimality. □

4.4. Normal axiomatizations

4.8. DEFINITION. An I-axiomatization $\mathcal{A}$ of $\mathcal{I}_E$ is *normal* if for all ground terms $s_1, \ldots, s_n$, and every predicate symbol P,

$$\left\{\begin{array}{rcl} s_1 \downarrow_{\mathcal{R}_E} \not\equiv s_2 \downarrow_{\mathcal{R}_E} & \text{implies} & \mathcal{A} \models s_1 \downarrow_{\mathcal{R}_E} \not\equiv s_2 \downarrow_{\mathcal{R}_E} \\ E \not\models P(s_1 \downarrow_{\mathcal{R}_E}, \ldots, s_n \downarrow_{\mathcal{R}_E}) & \text{implies} & \mathcal{A} \models \neg P(s_1 \downarrow_{\mathcal{R}_E}, \ldots, s_n \downarrow_{\mathcal{R}_E}) \end{array}\right.$$

Let us give three examples of normal axiomatizations which can be obtained using the techniques described in section 5:

4.9. EXAMPLE. Back to example 1.1, the axiomatization given in example 3.1 is normal.

4.10. EXAMPLE. Consider

$$\mathcal{A} = \{s = t \Rightarrow s \equiv t \mid s, t \in T(F), N(s), N(t)\}$$

where N and $\equiv$ are the predicates defined on $T(F)$ by the following rules:

$$\left\{\begin{array}{ll} P(0) & \\ P(s(x)) \Leftarrow P(x) & \\ N(x) \Leftarrow P(x) & \\ N(-x) \Leftarrow P(x) & \\ f(s_1, \ldots, s_n) \equiv f(t_1, \ldots, t_n) \Leftrightarrow s_1 = t_1 \wedge \ldots \wedge s_n = t_n & \text{for every } f \in F \\ f(s_1, \ldots, s_n) \not\equiv g(t_1, \ldots, t_m) & \text{for every } f \neq g \end{array}\right.$$

$\mathcal{A}$ is a normal axiomatization for the example 1.2. Intuitively, the new predicate N is satisfied on ground terms in normal form w.r.t. the system of example 2.9.

4.11. EXAMPLE. $F = \{0, s\}$ with one predicate symbol $>$.

$$E = \left\{\begin{array}{l} s(x) > 0 \\ s(x) > s(y) \Leftarrow x > y \end{array}\right.$$

is supposed to define the strict ordering on natural numbers.

$$\mathcal{A} = \left\{\begin{array}{l} \neg(0 > x) \\ \neg(s(x) > s(y)) \Leftarrow \neg(x > y) \end{array}\right.$$

is a normal axiomatization of $\mathcal{I}_E$.

If we let $\mathcal{S}_N$ be defined by the rules of the previous section, then the main result is the following:

4.12. THEOREM. *If E is a finite saturated set of Horn clauses and if $\mathcal{A}$ is a normal axiomatization, then $\mathcal{S}_N$ is an inductive saturation strategy.*

PROOF. (sketch). Assuming that $\mathcal{A} \cup E \cup \mathcal{C}$ is inconsistent and that $\mathcal{S}_N(E, \mathcal{C}) = \emptyset$, we will prove that there is a clause $c \in \mathcal{C}$ and a substitution σ such that $c\sigma \cup \mathcal{A}$ is inconsistent. By lemma 3.5, we know that there is at least one c and one σ such that $c\sigma \cup E \cup \mathcal{A}$ is inconsistent. We choose them in such a way that $c\sigma$ is minimal among ground instances of clauses in $\mathcal{C}$ which are inconsistent with $E \cup \mathcal{A}$. Then, for every literal L of $c\sigma$, $\mathcal{I}_E \not\models L$. Since $\mathcal{S}_N(E, \mathcal{C})$ is redundant, all inferences $c, E \vdash c'$ are redundant. Hence, by lemma 4.7, $c\sigma$ is irreducible by $\mathcal{R}_E$: all terms occurring in L are irreducible by $\mathcal{R}_E$. We investigate all possibilities for L:

- if L is a literal $P(s_1, \ldots, s_n)$, $E \not\models L$ by hypothesis, hence $\mathcal{A} \models \neg L$ by definition and $\mathcal{A} \cup L$ is inconsistent
- if L is a literal $\neg P(s_1, \ldots, s_n)$, by lemma 4.7, and since $\mathcal{S}_N(E, \mathcal{C}) = \emptyset$, $E \not\models P(s_1, \ldots, s_n)$, hence $\mathcal{A} \models L$; this case cannot occur.
- if L is an equation $s = t$, by definition of $\mathcal{A}$, $s = t \cup \mathcal{A}$ is inconsistent
- if L is a negated equation $s \neq t$, then $\mathcal{I}_E \models s = t$ and, since they are both in normal form, $s \equiv t$. Hence L alone is inconsistent.

$\square$

4.5. Examples

For historical reasons, in most examples of the proof by consistency method, the set of axioms is a set of equations which can be turned into a ground convergent rewrite system. A ground convergent rewrite system is actually a saturated set of clauses. Instead of redundancy, one uses so-called simplification and deletion rules, which are particular instances of redundancy criteria. The deduction rules in this case consist in critical pairs computation, which again is a particular case of superposition.

4.13. EXAMPLE. Let us go on with the toy example 1.1. We choose for $\succeq$ for instance the lexicographic path ordering extending the precedence $+ > s > 0$. E is turned into a convergent set of rules:

$$\begin{cases} (1) & 0 + x \rightarrow x \\ (2) & s(x) + y \rightarrow s(x + y) \end{cases}$$

hence saturated. We are going to consider 4 different conjectures which illustrate different facets of the deduction methods.

Consider the conjecture

$$(c_1) \qquad x + y = y + x$$

There are four possible inferences, overlapping (c_1) with (1) or (2). We get first

$$\begin{array}{lrcl} (c_{1,1}) & x + 0 & \rightarrow & x \\ (c_{1,2}) & y + s(x) & \rightarrow & s(x + y) \end{array}$$

the arrow indicating an ordering between the two members of the equations. The two other inferences yield renamings of the two above equations, which are therefore redundant. Overlapping E on the two new conjectures, we get 4 new conjectures:

$$\begin{array}{lrcl}
(c_{1,3}) & 0 & = & 0 \\
(c_{1,4}) & s(x+0) & \to & s(x) \\
(c_{1,5}) & s(x+0) & \to & s(x) \\
(c_{1,6}) & s(y+s(x)) & = & s(x+s(y))
\end{array}$$

The four new clauses are all redundant. This is obvious for $(c_{1,3})$. $(c_{1,4})$ and $(c_{1,5})$ are identical. Hence we only have to perform two redundancy proofs. Consider any ground substitution σ. $(c_{1,4}, \sigma) \gg (c_{1,1}, \sigma)$ and $c_{1,1}\sigma \models c_{1,4}\sigma$, which proves the redundancy of $(c_{1,4})$. Now $(c_{1,6}, \sigma) \gg (c_{1,2}, \sigma), (c_{1,2}, \sigma'), (c_1, \sigma'')$ where $\sigma' = \{x \mapsto y\sigma; y \mapsto x\sigma\}$ and $\sigma'' = \sigma$. Moreover,

$$s(y+s(x))\sigma \underset{c_{1,2}}{\overset{\sigma}{\longleftrightarrow}} s(s(x+y))\sigma \underset{c_1}{\overset{\sigma''}{\longleftrightarrow}} s(s(y+x)) \underset{c_{1,2}}{\overset{\sigma'}{\longleftrightarrow}} s(x+s(y))\sigma$$

which shows that $c_{1,2}\sigma, c_{1,2}\sigma', c_1\sigma'' \models c_{1,6}\sigma$, hence $c_{1,6}$ is redundant. This redundancy proof can also be seen as simplifying the two members of $(c_{1,6})$ using $c_{1,2}$, then using c_1 on a subterm.

Hence $\{c_1, c_{1,1}, c_{1,2}\} \cup E$ is saturated. We only have to check that each of these clauses is consistent with $\mathcal{A}$ (see example 3.1), which is the case.

Consider now the conjecture

$$(c_2) \qquad x+y=0 \Rightarrow x=0$$

As before, overlapping E into c_2 yields two clauses

$$\begin{array}{lrcl}
(c_{2,1}) & y=0 & \Rightarrow & 0=0 \\
(c_{2,2}) & s(x+y)=0 & \Rightarrow & s(x)=0
\end{array}$$

The clause $c_{2,1}$ is a tautology, hence redundant. The clause $c_{2,2}$ is not redundant, according to our definition. Then the saturation process will go on forever, generating the clauses:

$$(c_{2,2n}) \qquad s^n(x+y)=0 \Rightarrow s^n(x)=0$$

This could have been avoided, noticing that $\mathcal{A} \models (c_{2,2})$. Such redundancy criteria could be incorporated in the method. However, the example shows some limitations, as it does not succeed on this simple example.

Consider now the conjecture

$$(c_3) \qquad x=0 \Rightarrow x+y=0$$

Overlapping E into (c_3) we get the two clauses:

$$\begin{array}{lrcl}
(c_{3,1}) & 0=0 & \Rightarrow & y=0 \\
(c_{3,2}) & s(x)=0 & \Rightarrow & s(x+y)=0
\end{array}$$

$(c_{3,2})$ is redundant (using $c_{3,1}$), as well as c_3; $E \cup c_{3,1}$ is saturated. But, of course, $c_{3,1} \cup \mathcal{A}$ is inconsistent, as can be seen after one resolution step. Hence c_3 is not an inductive theorem.

Consider now the conjecture

$$(c_4) \qquad x \neq s(x)$$

There is not any possible inference rule here: $E \cup (c_4)$ is saturated. Since c_4 is consistent with $\mathcal{A}$, c_4 is valid in $\mathcal{I}_E$. (It is not an inductive theorem however, as this is a negative conjecture).

4.6. Further refinements

4.6.1. Selection strategies

It is possible to further refine the strategy which has been presented in section 4.3. For instance, we may use a fair selection function among maximal atoms. Let us illustrate this with an example.

4.14. EXAMPLE. We continue the example 4.11. A classical conjecture is the transitivity of the ordering:

$$(c_1) \qquad x > y \wedge y > z \Rightarrow x > z$$

The selection strategy marks the maximal negative literals when none of them is marked. Then one of the marked literals is selected. Marking is inherited by non-selected literals along inferences. This refinement is compatible with the previous strategy: lemma 4.7 still holds with this additional restriction.

(c_1) contains two maximal negative literals which are marked:

$$(c_1) \qquad \underline{x > y} \wedge \underline{y > z} \Rightarrow x > z$$

We choose one (and only one) of them for the inference. We get (by resolution) two new conjectures:

$$\begin{array}{llcl} (c_{1,1}) & \underline{0 > z} & \Rightarrow & s(x) > z \\ (c_{1,2}) & x > y \wedge \underline{s(y) > z} & \Rightarrow & s(x) > z \end{array}$$

There is not any possible deduction with $c_{1,1}$. E can be overlapped on $c_{1,2}$ yielding a third conjecture:

$$(c_{1,3}) \qquad x > y \wedge y > z \Rightarrow s(x) > s(z)$$

This last clause is redundant, using (c_1) and the second axiom of E. Then $E \cup \{c_1, c_{1,1}, c_{1,2}\}$ is saturated. It remains to check consistency with $\mathcal{A}$, which is even simpler.

Note that, on this example, without a selection strategy, considering always the overlap with the leftmost literal, we would enter an infinite loop generating the clauses

$$x > y \wedge s^n(y) > z \Rightarrow s^n(x) > z$$

4.6.2. Complete sets of positions

It is also possible to restrict the overlaps that have to be considered to *inductively complete positions* ([Fribourg 1989]).

4.15. DEFINITION. A position p in a term t is *inductively complete* if, for every ground substitution σ which is irreducible w.r.t. $\mathcal{R}_E$, the subterm of $t\sigma$ at position p is reducible by $\mathcal{R}_E$.

4.16. EXAMPLE. Let us consider yet another conjecture following example 4.13.

$$(c_5) \qquad (x+y)+z = x+(y+z)$$

This conjecture can be proved without any refinement of the inductive completion strategy if the status of $+$ is chosen from left to right. Suppose however that we did a mistake and that we chose a status right-left. Then there are a priori two possible overlaps on $x+(y+z)$. However, we have seen that $+$ is a defined symbol, hence every position of a $+$ is inductively complete. Then we may consider either of the two positions: the topmost one or position 2 for the superposition. But both positions do not have to be considered. This yields in few steps a saturated set of formulas, proving that c_5 is an inductive theorem.

Assuming the following axioms defining $*$:

$$\begin{cases} 0 * x &= 0 \\ s(x) * y &= y + x * y \end{cases}$$

the saturation strategy succeeds in proving the following conjectures:

$$\begin{cases} x * (y+z) &= x * y + x * z \\ x * y &= y * x \\ x * (y * z) &= (x * y) * z \end{cases}$$

as well as a few other extensions, as described in [Bachmair 1991].

More generally, it is possible to restrict the overlaps to *inductively complete sets of positions*, as shown in [Küchlin 1987]. A set $\{p_1, \ldots, p_n\}$ is an inductively complete set of positions in t if for every irreducible substitution σ one of the subterms of $t\sigma$ at positions $p_1, \ldots, p_n$ is reducible.

4.6.3. Non-saturated sets of axioms

Another generalization consists in using built-in theories such as associativity and commutativity . The deduction system can be modified, taking into account such theories, along the lines described in chapter XVII. The main difficulty is to design a normal axiomatization in this context.

Finally, the main remaining restriction is the hypothesis on saturatedness of E. This hypothesis can be weakened too [Gramlich 1990*b*, Comon and Nieuwenhuis 1998].

5. Examples of Axiomatizations $\mathcal{A}$ from the Literature

Up to know, we have set up a very general framework for the proof by consistency (or *inductionless induction*) method. However, historically, the method was not explained in this way. The method was actually designed by implicitly (I mean: this was hidden in the inconsistency detection rules) assuming some specific I-axiomatization. In this section, some examples of proof by consistency techniques which can be found in the literature are given. For each of these examples, we will make explicit the (normal) I-axiomatization $\mathcal{A}$.

Each example is characterized by

1. the hypotheses on E
2. the (normal) I-axiomatization
3. the deduction method (not always an inductive saturation strategy)
4. the decision technique for checking the consistency of $\{s = t\} \cup \mathcal{A}$

5.1. Musser's approach [Musser 1980]

At about the same time (1980), D. Musser [Musser 1980] and J. Goguen [Goguen 1980] proposed the following method.

F is assumed to contain a particular function symbol eq and two constants $true$ and $false$. (Actually, we would need a particular *sort* for the Booleans, but we skip this irrelevant aspect here). E is a finite set of equations and it is assumed that, for every ground terms s, t

$$\begin{array}{rcl} s =_E t & \Leftrightarrow & eq(s,t) =_E true \\ s \neq_E t & \Leftrightarrow & eq(s,t) =_E false \end{array}$$

That is, eq is "completely defined".

Then, running a completion procedure, a contradiction (and therefore a disproof) is detected when an equation $true = false$ is generated. On the other hand, $s = t$ is an inductive consequence of E if $E \cup \{s = t\}$ can be completed into a convergent rewrite system without generating the inconsistent equation $true = false$. This approach can be seen as imposing an equational I-axiomatization, which is part of E itself. Then $\mathcal{A}$ simply reduces to $\{true \neq false\}$,

Actually, these results are straightforward. Indeed, choosing $\mathcal{A}$, the equational deduction method and the inconsistency detection as in figure 1, they are consequences of the following proposition:

5.1. PROPOSITION. *Under the above assumptions, $\mathcal{A}$ is an I-axiomatization.*

PROOF. By hypothesis, $true \neq_E false$, therefore $\mathcal{I}_E \models \mathcal{A}$. Now, let $\mathcal{M}$ be an Herbrand model of E and s, t be two ground terms. There are four cases:

1. $\mathcal{M} \models s = t$ and $s =_E t$

1. E is a finite convergent rewrite system such that $s =_E t$ iff $eq(s,t) =_E true$ and $s \neq_E t$ iff $eq(s,t) =_E false$.
2. $\mathcal{A} = \{true \neq false\}$
3. The deduction engine is given by the Knuth-Bendix completion (assuming it does not fail) and any reduction ordering in which $true$ and $false$ are smaller than the other function symbols.
4. Inconsistency detection is given by the only rule:

$$true = false, true \neq false \vdash \Box$$

Figure 1: Musser's Approach

2. $\mathcal{M} \not\models s = t$ and $s \neq_E t$
3. $\mathcal{M} \not\models s = t$ and $s =_E t$. In this case $\mathcal{M} \not\models E$.
4. $\mathcal{M} \models s = t$ and $s \neq_E t$. In this case, $eq(s,t) =_E false$. Therefore, either $\mathcal{M} \not\models E$ or $\mathcal{M} \models eq(s,t) = false$. In the latter case, $\mathcal{M} \models true = eq(s,s) = eq(s,t) = false$ and therefore $\mathcal{M} \not\models \mathcal{A}$.

As a conclusion, if $\mathcal{M} \models E \cup \mathcal{A}$, then, for all ground terms s, and t. $s =_E t$ iff $\mathcal{M} \models s = t$. $\Box$

Note that the axiomatization is not normal since, for two ground terms s, t such that $s \neq_E t$, we don't necessarily have $true \neq false \models s \downarrow_{\mathcal{R}_E} \neq t \downarrow_{\mathcal{R}_E}$. This can be recovered as follows: let $\mathcal{D}$ be the equations defining eq. Remove $\mathcal{D}$ from E and let $\mathcal{A} = \mathcal{D} \cup \{s \neq t \Leftrightarrow eq(s,t) = false\}$. Then $\mathcal{A}$ is a normal I-axiomatization. This is more in the spirit of Musser's technique as, roughly, $\mathcal{D}$ is an equational I-axiomatization.

5.2. EXAMPLE. Consider again example 1.1. The following set of rules completely defines eq:

$$\left\{\begin{array}{rcl} eq(0, s(x)) & \to & false \\ eq(s(x), 0) & \to & false \\ eq(s(x), s(y)) & \to & eq(x, y) \\ eq(x, x) & \to & true \end{array}\right.$$

Then the method will disprove e.g. $s(x) + 0 = x$ as follows: add $s(x) + 0 = x$ to the set of rules. This gives the rewrite system:

$$\left\{\begin{array}{rcll} 0+x & \rightarrow & x & (1)\\ s(x)+y & \rightarrow & s(x+y) & (2)\\ eq(0,s(x)) & \rightarrow & false & (3)\\ eq(s(x),0) & \rightarrow & false & (4)\\ eq(s(x),s(y)) & \rightarrow & eq(x,y) & (5)\\ eq(x,x) & \rightarrow & true & (6)\\ s(x)+0 & \rightarrow & x & (7) \end{array}\right.$$

Simplifying (7) with (2), (7) is replaced with

$$s(x+0) \rightarrow x \qquad (8)$$

Overlapping (1) and (8) we get a new rule

$$s(0) \rightarrow 0 \qquad (9)$$

Overlapping (4) and (9) we get

$$eq(0,0) \rightarrow false \qquad (10)$$

Overlapping (6) and (9) we get finally $true = false$: the system is inconsistent and hence the conjecture is false.

Now, consider the conjecture $x+0 = x$. Completion of the above system, replacing the conjecture $s(x)+0 \rightarrow x$ with the new conjecture $x+0 \rightarrow x$ yields the convergent system:

$$\left\{\begin{array}{rcl} 0+x & \rightarrow & x\\ s(x)+y & \rightarrow & s(x+y)\\ eq(0,s(x)) & \rightarrow & false\\ eq(s(x),0) & \rightarrow & false\\ eq(s(x),s(y)) & \rightarrow & eq(x,y)\\ eq(x,x) & \rightarrow & true\\ x+0 & \rightarrow & x \end{array}\right.$$

$true$ and $false$ are still distinct, hence the conjecture is consistent with $\mathcal{A}$: it is an inductive theorem.

Now, of course, this approach is much too restrictive in many respects. For example, the requirement that an equality predicate is completely specified is much too strong:

5.3. EXAMPLE. Assume that F consists of 0 (constant), s, p (unary) and E contains the two equations

$$\left\{\begin{array}{rcl} s(p(x)) & = & x\\ p(s(x)) & = & x \end{array}\right.$$

1. C is a set of free constructors. E is a finite convergent rewrite system.
2.
$$\begin{array}{rl} \mathcal{A} = & \{f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n) \Rightarrow x_1 = y_1 \wedge \ldots \wedge x_n = y_n \mid f \in C\} \\ \cup & \{f(x_1,\ldots,x_n) \neq g(y_1,\ldots,y_m) \mid f,g \in C, f \neq g\} \end{array}$$
3. equational deduction is given by the Knuth-Bendix completion procedure in which constructor (ground) terms are assumed to be smaller than any non-constructor term.
4. Inconsistency detection consists of using $\mathcal{A}$ to decompose equations headed with constructor symbols.

Figure 2: Huet and Hullot's approach

It is not possible to design a finite set of equations E' which completely defines eq (in the above sense). (This is left as an exercise).

Another weakness is the use of standard completion which may fail: this deduction system is not an inductive saturation strategy. It is not refutationally complete.

5.2. Huet and Hullot's method [Huet and Hullot 1982]

In their 1982 paper Huet and Hullot proposed another variant of the method. They assume that the set of function symbols F contains a subset C of free constructors. The main components of the method are summarized in figure 2. Such a formulation was proposed by L. Fribourg as early as 1984 in [Fribourg 1984].

For example, compare the set $\mathcal{A}$ above with the set given in example 3.1.

5.4. PROPOSITION. *Under the above hypotheses, $\mathcal{A}$ is a normal I-axiomatization.*

PROOF. The recursiveness of $\mathcal{A}$ is obvious. Assume that $x_1,\ldots,x_n.y_1,\ldots,y_m \in T(F)$ and $f(x_1,\ldots,x_n) =_E f(y_1,\ldots,y_n)$ for some $f \in C$. We may actually assume that $x_1,\ldots,x_n,y_1,\ldots,y_n \in T(C)$ because, by hypothesis, C is a set of constructors, hence each term in $T(F)$ can be proved to be equal to a term in $T(C)$. Then, $f(x_1,\ldots,x_n)$ and $f(y_1,\ldots,y_n)$ are in $T(C)$. Since the constructors are free, they are equal iff they are identical. This shows that the first set of axioms in $\mathcal{A}$ is satisfied in $T(F)/_{=_E}$. The verification is similar for the second set of axioms: $\mathcal{I}_E \models \mathcal{A}$.

Assume now that $\mathcal{M}$ is an Herbrand model of $\mathcal{A} \cup E$. Let $s,t \in T(F)$. Since $\mathcal{M} \models E$, and by definition of a set of constructors, there are two terms $s' \equiv f(s_1,\ldots,s_n)$ and $t' \equiv g(t_1,\ldots,t_m)$ such that $s',t' \in T(C)$ and $\mathcal{M} \models s = s' \wedge t = t'$. And, since $\mathcal{M} \models \mathcal{A}$, s' and t' must be identical as soon as s and t are equal. This means that $\mathcal{M} \models s = t$ implies $E \vdash s = t$: $\mathcal{M}$ is isomorphic to $\mathcal{I}_E$.

$\mathcal{A}$ is normal since constructor terms are irreducible. □

5.5. EXAMPLE. Let us consider again the toy example 1.1. $x+0=x$ is an inductive theorem since, adding the rule $x+0 \to x$ to the two original rules $0+x \to x$ and $s(x)+y \to s(x+y)$ we get a convergent term rewriting system which is consistent with $\mathcal{A}$.

The next example was given by Huet and Hullot [Huet and Hullot 1982]. They were able to show for instance that $rev(rev(x)) = x$ automatically, without introducing new lemmas, which was required in explicit induction techniques (typically one needs the associativity of @).

5.6. EXAMPLE. The set of function symbols consists of $0, s, +, nil, cons, @, rev, \mathsf{length}$ We have to assume a *sort structure* (which was not introduced so far for the sake of simplicity): $0, s, +$ take only integers as arguments and return integers. $nil, @, rev$ take only lists as arguments and return lists. *cons* takes one integer and a list and returns a list. L takes a list as argument and returns an integer.

$$\begin{cases} (1) & 0+x & \to & x \\ (2) & s(x)+y & \to & s(x+y) \\ (3) & nil@x & \to & x \\ (4) & cons(n,x)@y & \to & cons(n,x@y) \\ (5) & \mathsf{length}(nil) & \to & 0 \\ (6) & \mathsf{length}(cons(n,x)) & \to & s(\mathsf{length}(x)) \\ (7) & rev(nil) & \to & nil \\ (8) & rev(cons(n,x)) & \to & rev(x)@cons(n,nil) \end{cases}$$

The specification is sufficiently complete w.r.t. $\{+, @, rev, \mathsf{length}\}$

Using the method described above, we can automatically prove the conjectures $\mathsf{length}(x@y) = \mathsf{length}(x) + \mathsf{length}(y))$, $\mathsf{length}(rev(x)) = \mathsf{length}(x)$, $rev(rev(x)) = x$

Actually the two approaches (Musser's approach and Huet and Hullot's approach) are equivalent:

5.7. LEMMA. *Huet and Hullot's conditions are equivalent to Musser's conditions. In other words, any specification containing a set of free constructors can be completed to a conservative extension in which the equality predicate is completely defined.*

Remarks

1. Though not necessary, it is also possible to use $\mathcal{A}$ in the completion process. This remark applies for other approaches as well. Actually, in their paper, Huet and Hullot add the following rule to the completion rules:

$$E \cup \{c(t_1, \ldots, t_n) = c(u_1, \ldots, u_n)\}; R \quad \Rightarrow \quad E \cup \{t_1 = u_1, \ldots, t_n = u_n\}; R$$

if $c \in C$. Which corresponds to the use of an axiom of $\mathcal{A}$.

2. In view of the examples 1.2 and 5.3, this approach is still too restrictive. We will see in the next section how it can be extended.
3. Another weakness is again the use of standard completion which may fail.

5.3. Jouannaud and Kounalis approach [Jouannaud and Kounalis 1989]

We show here a slight extension of the Jouannaud and Kounalis approach. Now, we do not assume any set of free constructors nor any equality predicate. Instead, we assume that the axioms of the specifications are oriented into a finite (ground) convergent rewrite system $\mathcal{R}_0$. Examples 3.1 and 2.7 are easy to complete in order to satisfy this condition.

$\mathcal{A}$ contains all disequalities $s \neq t$ for terms s, t such that $s \succ t$ and s is not reducible. $\mathcal{A}$ is, a priori, infinite. Using it in an effective way relies on the key notion of *ground reducibility.*

5.8. DEFINITION. A term t is *ground reducible* (w.r.t. $\mathcal{R}_0$) if all its ground instances are reducible.

Let *Red* be the ground reducibility predicate, which holds true on a term s when all ground instances of s are reducible. *Red* can be defined using a finite set of Horn clauses. As shown in section 5.5 below, $\neg$*Red* can also be defined by a finite set of Horn clauses $\mathcal{S}_{\neg Red}$.

It is not very difficult to see that an equation $s = t$ such that $s \succ t$ is consistent with $\mathcal{A}$ if and only if $Red(s)$ is consistent with $\mathcal{S}_{\neg Red}$. Such a consistency test can be perfomed by any first-order theorem prover. It turns out however that *Red* is recursive (in other words, the consistency of $\{Red(s)\} \cup \mathcal{S}_{\neg Red}$ is decidable), but this is quite difficult to prove and section 6 is devoted to this question and the feed-back on some (other) possible axiomatizations. For the moment, we see *Red* as a black box and assume that $\mathcal{A}$ is a recursive set of purely universal formulas indeed.

The main components of the methods are described in figure 3.

5.9. PROPOSITION. *$\mathcal{A}$ is a normal I-axiomatization.*

PROOF. Assume that for all ground substitution σ, $s\sigma =_E t\sigma$. Then, because $\mathcal{R}_0$ is convergent, $s\sigma\downarrow_{\mathcal{R}_0} \equiv t\sigma\downarrow_{\mathcal{R}_0}$ which shows that either $s\sigma$ is reducible or $t\sigma$ is reducible or the two terms are syntactically equal. If, $s\sigma \succ t\sigma$, we are necessary in the first case: $s\sigma$ is reducible. Hence $\mathcal{I}_E \models \mathcal{A}$.

Let E be the set $\mathcal{R}_0$ where the rules are considered as (unoriented) equations. Assume $T(F)/_\sim$ satisfies $\mathcal{A} \cup E$. Let s, t be two ground terms such that $s \sim t$. We prove by induction w.r.t. the reduction ordering $\to^*_{\mathcal{R}_0}$ on s, t that $s =_E t$. If s, t are irreducible w.r.t $\to_{\mathcal{R}_0}$, then, by $\mathcal{A}$, s and t must be identical, hence $s =_E t$. Now, assume that s or t is reducible. For example $s \to_{\mathcal{R}_0} s_1$. Then $s_1 \sim t$ since $T(F)/\sim$ satisfies E. By induction hypothesis, $s_1 =_E t$, hence $s =_E t$.

1. E is given by a finite ground convergent rewrite system $\mathcal{R}_0$.
2. $$\mathcal{A} = \{s \neq t \mid s,t \in T(F), s \succ t, \neg Red(s)\}$$
3. Equational deduction is given by the Knuth-Bendix completion procedure.
4. Inconsistency detection is given by the ground reducibility test.

Figure 3: Jouannaud and Kounalis approach [Jouannaud and Kounalis 1986]

By definition of $\mathcal{A}$, it is normal: two distinct ground terms are comparable w.r.t. $\succ$ and, if they were equal, this would imply the reducibility of the largest one. $\square$

5.10. EXAMPLE. The following example cannot be proved directly by explicit induction without introducing new lemmas. It can however be automatically proved by the inductionless induction method. As it can be easily seen, it does not fit in Huet and Hullot's approach.

We consider the following axioms

$$\begin{cases} (1) & 0+x & = & x \\ (2) & s(x)+y & = & s(x+y) \\ (3) & s(s(0)) & = & 0 \end{cases}$$

Let us show for instance that $x+x=0$ is an inductive consequence of the axioms. First, we have to fulfill the requirements, hence to complete the system (1),(2),(3) into a convergent rewrite system. We get

$$(4)\;\; s(s(x)) \;\; \rightarrow \;\; x$$

by overlapping (2) and (3), which subsumes (3). Then the system is convergent. Next, we add the conjecture

$$(5)\;\; x+x \;\; \rightarrow \;\; 0$$

Overlapping (5) and (2) we get

$$(6)\;\; s(x+s(x)) \;\; \rightarrow \;\; 0$$

Overlapping (4) and (6), we get

$$(7)\;\; x+s(x) \;\; \rightarrow \;\; s(0)$$

which simplifies (6). Then (1),(2),(4),(5),(7) is a convergent system. Moreover, $x+x$ and $x+s(x)$ are ground reducible (i.e. they satisfy the predicate *Red* for all values of x; this result should be assumed until we see how it is proved automatically in section 6), hence this system is compatible with $\mathcal{A}$: this shows that $x + x = 0$ is indeed an inductive theorem.

Now, consider the conjecture $x + s(x) = 0$. Overlapping $x + s(x) \to 0$ and (2) we get, after simplification, $s(0) = 0$ which yields a contradiction since $s(0)$ is not ground reducible: $x + s(x) = 0$ is not an inductive theorem.

The following example is a bit more complex. It describes a group generated by a, b where a has order 3 and b order 2. It is not possible to define a set of free constructors, since $*$ is not free and it occurs in all terms of some equivalence classes.

5.11. EXAMPLE. Function symbols are constants a, b, e, the binary symbol $*$ and the unary symbol $^{-1}$.

$$\left\{\begin{array}{rcl@{\qquad}rcl}
e^{-1} & \to & e & a^{-1} & \to & a * a \\
e * x & \to & x & x * e & \to & x \\
(x^{-1})^{-1} & \to & x & x^{-1} * x & \to & e \\
x * x^{-1} & \to & e & (x * y)^{-1} & \to & y^{-1} * x^{-1} \\
(x * y) * z & \to & x * (y * z) & a * (a * a) & \to & e \\
x^{-1} * (x * y) & \to & y & x * (x^{-1} * y) & \to & y \\
a * (a * (a * x)) & \to & x & b * b & \to & e
\end{array}\right.$$

The system is convergent and allows to disprove automatically, for instance $x * (x * (x * (x * (x * x)))) = e$ (i.e. every element has order 6) and gives a counter-example. Adding the axiom $a * (a * b) = b * a$, we may however prove that every element has order 6.

In this section, we gave only positive examples. There are however several weaknesses:

1. First we must meet the condition: the axioms should be turned into a convergent rewrite system. This is a strong restriction and many examples do not fall into this category.
2. Adding the conjecture, the completion may not terminate. This is unavoidable since, otherwise, the set of inductive theorems would be recursive.
3. We cannot prove unorientable conjectures such as commutativity since the standard completion fails in this case.

All these drawbacks actually come from the deduction system (Knuth-Bendix completion).

5.4. Bachmair's approach [Bachmair 1988]

The first refutationally complete inductive completion method was proposed by Fribourg in [Fribourg 1984]. Bachmair's method is the first one which does not

1. E is assumed to be given by a finite ground rewrite system $\mathcal{R}_0$.
2. $$\mathcal{A} = \{s \neq t \mid s,t \in T(F), \neg Red(s), s \succ t\}$$
3. Equational deduction is given by a linear restriction of the Knuth-Bendix completion procedure.
4. Inconsistency detection is given by the (co)-ground reducibility test.

Figure 4: Bachmair's approach

assume a constructor discipline and which is also refutationally complete, hence is an inductive saturation strategy, following our terminology.

Assumptions on the axioms are the same as in the previous section. The axiomatization is also the same. The differences lie in the proof strategies. As in section 4, only overlaps of axioms on conjectures are necessary and conjectures need not be oriented. This is referred to as a *linear strategy* and was first proposed in [Fribourg 1986]. The main ingredients of the method are given in figure 4.

Though the axiomatization is the same as in the previous section, it is not used exactly in the same way, however; for each generated equation $s = t$, either $s \succ t$ (resp. $t \succ s$), then the consistency of $\{s = t\} \cup \mathcal{A}$ reduces to the consistency of $\{Red(s)\} \cup \mathcal{S}_{\neg Red}$, as before. Or else s and t are incomparable in the ordering. In this case, the consistency of $s = t$ with $\mathcal{A}$ reduces to the consistency of the following finite set of clauses:

$$\begin{cases} Red(s) \vee Red(t) \vee s \equiv t \\ \mathcal{S}_{\neg Red} \\ \mathcal{S}_{\neg\equiv} \end{cases}$$

where $\mathcal{S}_{\neg\equiv}$ is a finite set of Horn clauses defining the negation of the syntactic equality on ground terms.

Now, in the equational deduction, conjectures need *not* to be oriented. We get something which looks like the inductive completion procedure of section 4.2 in the pure equational case: if an equation is orientable, then only superposition on the maximal side have to be considered. If the equation is not orientable, then superpositions on both sides have to be computed. This is the main idea of *unfailing completion* which is described in chapter XVIII.

Examples of this method are for instance the commutativity of addition in example 4.13 which cannot be handled by Jouannaud and Kounalis method.

Though Bachmair's method answer one of the limitations of the Jouannaud and Kounalis one (now the completion cannot fail any more), the two other limitations remain: the original set of axioms has to be ground convergent and the saturation may not terminate.

5.5. Further examples of normal axiomatizations

Other instances of the proof by consistency approach and normal axiomatizations computations are described in [Comon and Nieuwenhuis 1998]. Some of them do not assume the saturatedness of E. For instance, we may add to E any sufficiently complete definition E' of a new function symbol; if $\mathcal{A}$ is a normal axiomatization w.r.t. E, it remains a normal axiomatization w.r.t. E', provided that the ground terms containing the new function symbol are larger than the terms which do not contain it. The saturatedness of E is not essential for the normal axiomatization. It is important for the inductive completion strategy, as we have seen.

5.12. Example. Consider example 1.1 and assume that we enrich the specification with the definition of the new function symbol *gcd*:

$$E' = \left\{ \begin{array}{rcl} gcd(0, x) & = & x \\ gcd(x, 0) & = & x \\ gcd(x + y, x) & = & gcd(y, x) \\ gcd(x, x + y) & = & gcd(x, y) \end{array} \right.$$

Then

$$\mathcal{A} = \left\{ \begin{array}{l} 0 \neq s(x) \\ s(x) = s(y) \Rightarrow x = y \end{array} \right.$$

remains a normal axiomatization of $E \cup E'$.

In the case of Horn clauses without equality, if, in each clause, the body of the clause does not contain variables which do not occur in the head (this will be called hereafter the *variable occurrence condition*), then it is possible to compute a normal axiomatization, using quantifier elimination for the first-order theory of finite trees [Comon 1991]. Let us show it on a very simple example:

5.13. Example. Let E be

$$E = \left\{ \begin{array}{rcl} E(0). & & \\ E(s(s(x))) & \Leftarrow & E(x). \end{array} \right.$$

The program is rewritten as

$$E(x) \Leftarrow x = 0 \vee (\exists y. x = s(s(y)) \wedge E(y))$$

In the least fixed point of the definition, the converse implication also holds, which gives by negating the two members:

$$\neg E(x) \Leftarrow x \neq 0 \wedge (\forall y. x \neq s(s(y)) \vee \neg E(y))$$

and, by so-called disunification:

$$\neg E(x) \Leftarrow (x = s(0) \vee (\exists y. x = s(s(y)) \wedge \neg E(y))$$

which is turned into the normal axiomatization $\mathcal{A}$:

$$\mathcal{A} = \begin{cases} \neg E(s(0)). \\ \neg E(s(s(x))) \Leftarrow \neg E(x) \end{cases}$$

As an instance of this procedure, we can compute from a set of Horn clauses defining *Red*, a set of Horn clauses defining $\neg$*Red*; typically *Red* can be defined by:

$$\begin{array}{rcll} Red(s) & & & \text{for every left hand side } s \text{ of a rule in } \mathcal{R} \\ Red(f(s_1, \ldots, s_n)) & \Leftarrow & Red(s_i) & \text{for every } f \in F \text{ and every } i \end{array}$$

which satisfies the variable occurrence property.

6. Ground Reducibility

Ground reducibility is formally defined on page 31. This predicate is also referred to as *Red* in the previous section. Decision of ground reducibility implies the decision of consistency with the normal axiomatizations given in sections 5.3 and 5.4. Its role however goes beyond this simple application: first, it is strongly related to sufficient completeness (see definition 2.6). Next, the works which have been devoted to this property introduce some techniques which are used for other purposes. For instance, *test sets* are the basis of the mechanization of induction in SPIKE [Bouhoula and Rusinowitch 1995] and *cover sets* are the basis of the mechanization of induction in (a part of) RRL [Kapur and Zhang 1995]. *Ground normal form languages* and their recognizers play a role in specification analysis in ReDuX and in the design of new induction techniques [Jouannaud and Bouhoula 1997].

The decidability of ground reducibility was shown originally by D. Plaisted [Plaisted 1985] and several other algorithms have been proposed since (e.g. [Jouannaud and Kounalis 1989, Kapur, Narendran and Zhang 1987, Kapur, Narendran, Rosenkrantz and Zhang 1991, Kounalis 1992, Caron, Coquidé and Dauchet 1993, Comon and Jacquemard 1997]). We sketch here two of them: one is based on *tree automata with constraints* along the lines of [Caron et al. 1993]. The second one is based on test sets, along the lines of [Kapur et al. 1987, Kounalis 1992]. Both of these methods have some feed back on inductive proofs since they yield an explicit induction scheme.

The complexity of Plaisted's decision algorithm is very high (a tower of 7 exponentials). The other algorithms reduce this upper bound. In the general case, the problem, including the computation of a "counter-example" in case of non-ground reducibility, is at least doubly exponential. When we stick to the decision problem, it is EXPTIME-complete [Comon and Jacquemard 1997].

6.1. Automata techniques

In [Caron et al. 1993], the authors show a quite general result. Given a term t, let encomp_t be a unary predicate symbol which is interpreted as the set $[\![\mathrm{encomp}_t]\!]$ of terms encompassing t.

6.1. Theorem ([Caron et al. 1993]). *Given n terms $t_1, \ldots, t_n$, the first-order theory of $encomp_{t_1}, \ldots encomp_{t_n}$ is decidable.*

On the other hand, ground reducibility of t w.r.t. R can be expressed by a particular formula in this theory: assuming that $l_1, \ldots, l_n$ are the left hand sides of R, t is ground reducible iff

$$\forall x.\mathrm{encomp}_t(x) \Rightarrow (\mathrm{encomp}_{l_1}(x) \vee \ldots \vee \mathrm{encomp}_{l_n}(x))$$

is valid in the above interpretation.

The proof of this property relies, as in Büchi's theorem, on a correspondence between formulas and automata: the authors design a kind of tree automata such that $[\![\mathrm{encomp}_t]\!]$ is accepted by such an automaton. Then the closure properties of the corresponding class of languages and the decidability of emptiness leads to the desired property.

It is out of the scope of this chapter to give all details of this result. Nevertheless, we are going to give a flavour of a more restricted result, sufficient for ground reducibility, which has some consequences for inductive proofs. We first define *automata with disequality constraints*. They generalize classical (bottom-up) tree automata by allowing them to check disequalities between subtrees.

6.2. Definition. An *automaton with disequality constraints* (over a given finite alphabet F of function symbols) consists of a finite set of states Q, a subset Q_f of *final states* and a *transition relation*, which is a finite set of constrained rewrite rules of the form

$$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \ldots, x_n))$$

where $q_1, \ldots, q_n, q$ are states and c is a conjunction of *disequality constraints* written $p \neq q$ where p, q are strings of natural numbers. When the conjunction is empty, c is written $\top$ or simply omitted.

When every constraint c is $\top$, we get a classical bottom-up tree automaton (see e.g. [Comon, Dauchet, Gilleron, Lugiez, Tison and Tommasi 1997]).

The disequality constraints can be viewed as unary predicate symbols and are interpreted as follows: $t \models p \neq q$ iff p, q are positions of t and $t|_p \not\equiv t|_q$. A ground term $t \in T(F \cup Q)$ rewrites to u using the constrained rule $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \ldots, x_n))$ at position p_0 if t rewrites to u (at position p_0) using the unconstrained rule and, moreover, the subterm of t at position p_0 satisfies c. The reduction relation associated with a production rule P is written (as for rewriting) $\xrightarrow[P]{}$. The variables $x_1, \ldots, x_n$ are always irrelevant when displaying P, hence they are not reported.

6.3. DEFINITION. A ground term t is *accepted* by the automaton (Q, Q_f, P) if $t \xrightarrow[P]{*} q(t)$ for some $q \in Q_f$. The *language* $L(\mathcal{A})$ accepted by an automaton $\mathcal{A}$ is the set of ground terms that are accepted by $\mathcal{A}$.

6.4. EXAMPLE. Let $F = \{0, s, +\}$, $Q = Q_f = \{q_0, q_s, q_+\}$ and

$$P = \begin{cases} 0 & \rightarrow & q_0 \\ s(q_0) & \rightarrow & q_s \\ s(q_s) & \rightarrow & q_s \\ s(q_+) & \rightarrow & q_+ \\ q_s + q_s & \xrightarrow{1 \neq 2} & q_+ \end{cases}$$

The term $s(0)+s(s(0))$ is accepted by the automaton ($s(s(0))$ and $s(0)$ are accepted in state q_s and the last rule can be applied at the root since $s(s(0)) \not\equiv s(0)$), whereas $s(0) + s(0)$ or $(s(0) + s(s(0))) + s(0)$ are not accepted.

These automata may also be non-deterministic (which is not shown by the example).

Now, the main interest of such automata lies in the two properties given as theorems 6.5 and 6.8.

6.5. THEOREM. *For any rewrite system R, there is an automaton with disequality constraints $\mathcal{A}$ which accepts the set of irreducible ground terms.*

6.6. EXAMPLE. The automaton of example 6.4 accepts all ground terms that are irreducible w.r.t. the rewrite system

$$\mathcal{R} = \begin{cases} 0 + x & \rightarrow & x \\ x + 0 & \rightarrow & x \\ x + x & \rightarrow & 0 \\ (x + y) + z & \rightarrow & x + (y + z) \\ s(x) + (y + z) & \rightarrow & s(x + (y + z)) \end{cases}$$

6.7. EXAMPLE. Let $R_2 = \{x + x \rightarrow 0; 0 + x \rightarrow x; x + 0 \rightarrow x; s(s(0)) \rightarrow 0\}$, and the alphabet be $\{0, s, +\}$. The automaton which recognizes the set of ground normal forms is given by:

$$Q = Q_f = \{q_0, q_{x+y}, q_{s(x)}\}$$

$$
\begin{array}{rclcrcl}
0 & \rightarrow & q_0 & \qquad & s(q_0) & \rightarrow & q_{s(x)} \\
s(q_{x+y}) & \rightarrow & q_{s(x)} & & s(q_{s(x)}) & \rightarrow & q_{s(x)} \\
q_{s(x)} + q_{s(x)} & \xrightarrow{1 \neq 2} & q_{x+y} & & q_{s(x)} + q_{x+y} & \rightarrow & q_{x+y} \\
q_{x+y} + q_{s(x)} & \rightarrow & q_{x+y} & & q_{x+y} + q_{x+y} & \xrightarrow{1 \neq 2} & q_{x+y}
\end{array}
$$

if we do not consider the rules yielding q_r. Moreover, removing the non-accessible states, only two rules remain:

$$
0 \quad \rightarrow \quad q_0 \qquad\qquad s(q_0) \quad \rightarrow \quad q_{s(x)}
$$

6.8. THEOREM. *Given an automaton with disequality constraints $\mathcal{A}$, it is decidable whether $L(\mathcal{A})$ is empty, resp. finite.*

This theorem is a generalization of the classical emptiness decision problem of finite tree automata.

In general the algorithm is also more complex: it is polynomial in the number of states and exponential in the size of the constraints.

Now, let us show how this can be applied to ground reducibility.

6.9. PROPOSITION. *The set of ground instances of a linear term t is accepted by a finite tree automaton.*

Hence, in the linear case, it is easy to decide ground reducibility: compute the automaton $\mathcal{A}_{NF}$ accepting the set of irreducible ground terms. Compute the automaton $\mathcal{A}_t$ which accepts the set of ground instances of t. Intersect the two automata (this can be done in polynomial time using standard techniques, see e.g. [Comon et al. 1997]). Finally check the emptiness of the resulting automaton. Hence, once the automaton $\mathcal{A}_{NF}$ has been computed, we have:

6.10. PROPOSITION. *Ground reducibility of a linear term w.r.t. a given left linear rewrite system can be decided in polynomial time.*

Note however that the problem becomes EXPTIME-complete if $\mathcal{A}_{NF}$ is not given [Kapur et al. 1991].

In the case of non-linear terms, the problem can also be reduced to emptiness decision of automata with disequality constraints. In any case, it can be solved in deterministic exponential time [Comon and Jacquemard 1997] and that is the best we can do [Kapur et al. 1991].

6.2. Test sets

The basic idea of test sets is very simple: find a finite set of terms TS such that t is ground reducible iff all its instances obtained by replacing its variables by a term in TS are reducible.

Such test sets TS always exist (see [Kapur et al. 1987]), but they may be very large. How is it possible to compute the test sets? The naive computation is quite simple. We build a tree (called a *reducibility tree*). We call i-instance of a term t any ground instance $t\sigma$ of t such that, for every variable x of t, $x\sigma$ is irreducible.

Start with a tree which contains only one node (the root, which is also a leaf), labeled with $t = x$ (a variable). Then we repeat the following steps:

1. Select a leaf of the tree whose label t has reducible i-instances. (i.e. one of its non-variable subterms is unifiable with a left hand side of a rule). If there is no such leaf, then stop.
2. Select a variable x of t (t is not ground because none of the leaves are reducible). Consider the set $S(x,t)$ obtained by splitting x over the signature. More precisely, $S(x,t) = \{t\{x \to f(\vec{y})\} \mid f \in F\}$. Let $S'(x,t)$ be the subset of irreducible terms in $S(x,t)$.
3. If $S'(x,t)$ is empty, then remove the leaf labeled with t from the tree and remove the parents which have no more child by this transformation.
4. If $S'(x,t)$ is not empty, then add to the node labeled with t as many children as there are elements in $S'(x,t)$ and label them with the elements of $S'(x,t)$.

A test set is given by the leaves of the reducibility tree.

Of course, this simple way of proceeding is not terminating in general. It is however possible to give a bound B on the depth of the terms labeling a node of the reducibility tree. If, in the above algorithm, it is not possible to add a node labeled with a term of size smaller than B, then the computation stops.

An invariant of this computation is that the set of irreducible ground terms is always contained in the set of all i-instances of the leaves.

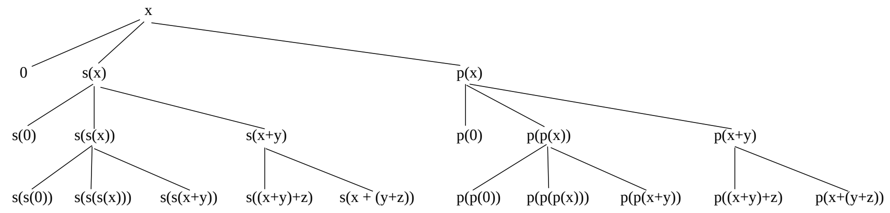
The bound B is really huge (from 5 to 7 exponentials depending on the version). It is however possible to restrict the computation in various ways. One of the main remark is the following: if, at some computation step, all leaves of a subtree labeled with t are labeled with terms strictly containing some instance of t then we can remove the whole subtree labeled with t. Indeed, assume that $t\sigma$ is a minimal (w.r.t. size) irreducible ground instance of t, then it should be an instance of some leaf of the subtree labeled with t, say $t\sigma \equiv u\theta$. However, by hypothesis, there is some strict subterm of u containing an instance of t. This means that there is an irreducible ground instance of t which is strictly smaller than $t\sigma$. Hence a contradiction.

6.11. EXAMPLE. Let us go back again to example 1.1. In order to compute a test set, we start with the variable x. Then split x into 0, $s(x)$ and $x + y$. The two first terms are not unifiable with a left hand side and are thus leaves of the reducibility tree. Remains $x+y$, which is split (say, according to the first variable). We get three terms $0 + y$, which is reducible, $s(x) + y$ which is reducible, and $(x + z) + y$. Then all descendants of $(x + y)$ contain an instance of $x + y$: this node can be removed.

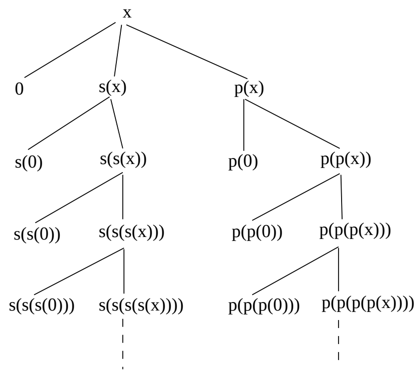
The reducibility tree now only contain one node and two leaves labeled respectively by 0 and $s(x)$. We got a test set containing two elements.

Test sets correspond to induction schemes and test sets computations can be seen as computing an induction scheme which was not explicit. There are however more complicated situations. Let us give a (still quite simple) example.

6.12. EXAMPLE. Consider example 2.7. Starting the computation of the reducibility tree, we get first $0, s(x), p(x), x + y$. 0 is irreducible and need not be further considered. The three other leaves are however unifiable with left hand sides. Thus, we have to go on adding new leaves; we get the following tree (after some steps):



The node labeled with $x + y$ has been removed according to the criterion: all its descendents contained some instance of it. Then, actually, it is possible to remove in the tree all terms containing $x + y$ (removing a node means that we already proved its ground reducibility). Now, we have the tree:



whose depth depend on the bound B.

It should be clear from example 6.12 that the test set is not "optimal". To clarify this point and have some feedback on the inductionless induction method, let us study the ground normal form languages.

6.3. Ground normal form languages

In the approaches of the last two sections, the aim was actually to find a description of the irreducible ground terms. In the first case we got a description via tree automata, while in the test set approach, we got a description using a set of i-instances of a test set.

Actually the test set method does not lead to a real description of irreducible ground terms but to an approximation which is sufficient for ground reducibility purposes only. That is why we stick now to the automata approach.

Assume we have an automaton $\mathcal{A}$ which accepts the set of irreducible ground terms and such that none of the states is useless (i.e. the corresponding language is not empty).

So far, we didn't consider type information. But it is of course possible to generalize the inductionless induction method to include typing information. More precisely, we may, instead of pure equational logic, consider a logic in which variables are constrained to range over certain tree languages called *sorts*.

6.13. EXAMPLE. Let us define *bool* and *int* as the tree languages accepted in the corresponding states q_{bool} and q_{int} of the automaton whose rules are

$$\begin{array}{rcl@{\qquad\qquad}rcl} 0 & \rightarrow & q_{int} & s(q_{int}) & \rightarrow & q_{int} \\ p(q_{int}) & \rightarrow & q_{int} & q_{int}+q_{int} & \rightarrow & q_{int} \\ E(q_{int}) & \rightarrow & q_{bool} & true & \rightarrow & q_{bool} \\ false & \rightarrow & q_{bool} & & & \end{array}$$

We will actually write *int* instead of q_{int} and *bool* instead of q_{bool}.

Then the following rules could be a specification of the function symbols:

$$\left\{ \begin{array}{rcl@{\qquad\qquad}l} s(p(x)) & \rightarrow & x & \mid \quad x \in int \\ p(s(x)) & \rightarrow & x & \mid \quad x \in int \\ 0+x & \rightarrow & x & \mid \quad x \in int \\ s(x)+y & \rightarrow & s(x+y) & \mid \quad x,y \in int \\ p(x)+y & \rightarrow & p(x+y) & \mid \quad x,y \in int \\ E(0) & \rightarrow & true & \\ E(s(0)) & \rightarrow & false & \\ E(s(s(x))) & \rightarrow & E(x) & \mid \quad x \in int \end{array} \right.$$

Deduction processes may be affected in general by these extensions. But it would lead us too far to consider again a proof system in this context. For the sake of simplicity, we will assume that the sort constraints always satisfy conditions under which exactly the same deduction rules as before are sound. (For example, the *sort decreasingness* property is a sufficient condition; see e.g. [Waldmann 1992]).

Now, introducing new sorts, corresponding to the states of the automaton $\mathcal{A}$, we may compute from the original specification another specification containing new sorts and new function symbols and which is equivalent to the original one, with additional properties. Here, by "equivalent" we mean that equalities built on the original signature are inductive consequences of the original set of axioms iff they are inductive consequences of the new set of axioms.

Let us illustrate the transformation on an example.

6.14. EXAMPLE. Consider the specification of example 2.7. It is not difficult to see that the automaton accepting the irreducible ground terms computed by the method described in section 6.1 is given by the transition rules:

$$\begin{array}{rclcrcl} 0 & \to & q_0 & \quad & s(q_0) & \to & q_{s(x)} \\ s(q_{s(x)}) & \to & q_{s(x)} & & p(q_0) & \to & q_{p(x)} \\ p(q_{p(x)}) & \to & q_{p(x)} & & & & \end{array}$$

We may now call q_0 *zero*, $q_{s(x)}$ *pos* and $q_{p(x)}$ *neg*. We also add a sort *int* containing all terms. We introduce two new function symbols s' and p' and claim that the following specification is equivalent to the original one.

$$\begin{array}{rclcrcl} 0 & \to & zero & \quad & s(int) & \to & int \\ s'(zero) & \to & pos & & p(int) & \to & int \\ s'(pos) & \to & pos & & int + int & \to & int \\ p'(zero) & \to & neg & & p'(neg) & \to & neg \end{array}$$

assuming moreover for sake of simplicity the ϵ-transitions $zero \to int$, $neg \to int$ and $pos \to int$. And the following set of axioms in addition to the original one:

$$\left\{ \begin{array}{rcllcl} s(x) & \to & s'(x) & & | & x \in pos \vee x \in zero \\ p(x) & \to & p'(x) & & | & x \in neg \vee x \in zero \\ s(p'(x)) & \to & x & & | & x \in neg \vee x \in zero \\ p(s'(x)) & \to & x & & | & x \in pos \vee x \in zero \\ s'(x) + y & \to & s(x+y) & & | & (x \in pos \vee x \in zero) \wedge y \in int \\ p'(x) + y & \to & p(x+y) & & | & (x \in neg \vee x \in zero) \wedge y \in int \end{array} \right.$$

How to compute this new set of axioms is beyond the scope of these notes (see [Comon 1989] instead).

Now, the main property of this new specification is that $\{0, s', p'\}$ is a set of free constructors! (We leave the proof as an exercise).

Hence it is possible to compute an equivalent (typed) specification for which there are now free constructors. This is not really surprising: the automaton provides the induction schemes that have to be followed in proving inductive properties.

But we cannot conclude directly that only the free constructor case has to be considered. Indeed, when the rewrite systems contain non-linear left hand sides, the typing rules may involve disequality constraints. What equational deduction become in such typed systems is unknown.

6.4. Sufficient Completeness

Sufficient completeness is used in areas other than automated deduction, for example to check that a function defined by pattern matching is indeed completely defined. We have also seen that in some situations, it is important to decide whether some subset of function symbols is a set of constructors.

Actually, sufficient completeness is strongly related to ground reducibility and that is why we add some words on this topic here. First, this property is undecidable (this was first shown in [Guttag and Horning 1978]), even in restricted situations:

6.15. PROPOSITION ([Kapur et al. 1987]). *It is undecidable whether a subset C of F is a set of constructors w.r.t. E, even when E is a finite convergent string rewrite system.*

But this can be recovered as follows:

6.16. PROPOSITION ([Jouannaud and Kounalis 1989]). *Let $\mathcal{R}$ be a convergent rewrite system for which normal forms of terms in $T(C)$ are in $T(C)$. Then C is a set of constructors iff, for each $f \in F \setminus C$, $f(x_1, \ldots, x_n)$ is ground reducible.*

Hence techniques that have been developed for ground reducibility apply for sufficient completeness as well: there is no need of further developments.

6.5. Limitations

The decidability of ground reducibility implies the decidability of the consistency of a conjecture with a normal axiomatization for the procedures described in sections 5.3 and 5.4, but what about other normal axiomatizations ?

Generalizing the construction of section 5.5 it is possible to build normal axiomatizations in several other situations, in particular when E is a finite set of Horn clauses with equality, or when E is a rewrite system modulo associativity and commutativity or even when E is a constrained rewrite system. However, the reducibility predicate is no longer recursive: ground reducibility is undecidable in the case of conditional rewrite systems, in the presence of associative-commutative symbols [Kapur et al. 1991] and for ordered rewriting [Comon, Narendran, Nieuwenhuis and Rusinowitch 1998]. Hence, in such situations, we need to use again a first-order theorem prover which is complete for refutation.

7. A comparison between inductive proofs and proofs by consistency

I express in this section personal opinions about the relationship between explicit (or implicit) induction and proof by consistency as described in this chapter.

Is the proof by consistency method more powerful ? Using an induction hypothesis on smaller instances than the instance we are currently considering is a particular case of the redundancy criterion which was defined in section 4. Hence, in principle, the inductive saturation strategies will terminate as soon as an explicit induction (using the same ordering) terminates. There are examples where the saturation process terminates whereas the explicit induction fails.

(That is because instances of the original conjecture need not be considered independently). As far as fully automated saturation is concerned, the proof by consistency method, at least in its first phase, is more powerful. It will succeed in all situations where the other method succeeds. Moreover, proof by consistency is a refutationally complete proof procedure. (This can also be obtained from an explicit induction procedure, as it is always possible to enumerate potential counter-examples. But the proof by consistency entirely relies on a as efficient as possible search for such counter-examples). Non-free constructors which cause additional problems in explicit induction techniques are irrelevant problems for the inductive saturation strategy. Finally, the proof by consistency method which was described here is also well-suited for proofs in $\mathcal{I}_E$, i.e. in the least fixed point of a set of Horn clauses.

Are explicit inductive proofs more powerful ? We should not draw conclusions too quickly from the previous paragraph. First, there is a second phase in the proof by consistency approach: we need to prove the consistency of each generated clause with $\mathcal{A}$. Though this phase is decidable in the equational case, it is EXPTIME-complete. In other cases, there is no precise evaluation of the price to pay for this step. Then, we may also have a different point of view; in the proof by consistency approach, roughly, the induction ordering is not free: it is the ordering which is used for the saturation of the axiom set.[4]

Finally, is "power" relevant in the context of inductive proofs ? In most situations described in Boyer and Moore book [Boyer and Moore 1979], the choice of relevant generalizations is crucial. This is not addressed at all in this chapter. The choice of an appropriate ordering is not addressed either. The main difficulties remain the same in both approaches.

Acknowledgments

I want to thank B. Gramlich, R. Nieuwenhuis, D. Sanella, D. Mc Allester, L. Fribourg and M. Rusinowitch for their comments on early versions of this chapter.

Bibliography

BACHMAIR L. [1988], Proof by consistency in equational theories, *in* 'Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh'.

BACHMAIR L. [1991], *Canonical Equational Proofs*, Birkhäuser, Boston.

BACHMAIR L. AND GANZINGER H. [1994], 'Rewrite-based equational theorem proving with selection and simplification', *Journal of Logic and Computation* **4**(3), 217–247.

BOUHOULA A. AND RUSINOWITCH M. [1995], 'Implicit induction in conditional theories', *Journal of Automated Reasoning* **14**(2), 189–235.

[4] It is claimed in [Kapur and Zhang n.d.] that a weakness of the proof by consistency approach is the need of a saturated set of axioms. I do not agree with this remark. See e.g. [Comon and Nieuwenhuis 1998] for more details on this.

BOYER R. S. AND MOORE J. S. [1979], *A computational logic*, ACM Monograph Series, Academic Press.

CARON A.-C., COQUIDÉ J.-L. AND DAUCHET M. [1993], Encompassment properties and automata with constraints, *in* C. Kirchner, ed., '5th International Conference on Rewriting Techniques and Applications', Vol. 690 of *Lecture Notes in Computer Science*, Springer-Verlag, Montreal, Canada.

COMON H. [1989], Inductive proofs by specifications transformation, *in* 'Proc. 3rd Rewriting Techniques and Applications, Chapel Hill, LNCS 355', Springer-Verlag, pp. 76–91.

COMON H. [1991], Disunification: a survey, *in* J.-L. Lassez and G. Plotkin, eds, 'Computational Logic: Essays in Honor of Alan Robinson', MIT Press.

COMON H., DAUCHET M., GILLERON R., LUGIEZ D., TISON S. AND TOMMASI M. [1997], Tree automata techniques and applications. A preliminary version of this unpublished book is available on `http://l3ux02.univ-lille3.fr/tata` .

COMON H. AND JACQUEMARD F. [1997], Ground reducibility is exptime-complete, *in* 'Proc. IEEE Symp. on Logic in Computer Science', IEEE Comp. Soc. Press, Varsaw.

COMON H., NARENDRAN P., NIEUWENHUIS R. AND RUSINOWITCH M. [1998], Decision problems in ordered rewriting, *in* 'Proc. IEEE Symp. Logic in Computer Science', Indianapolis.

COMON H. AND NIEUWENHUIS R. [1998], Induction = i-axiomatization + first-order consistency, Research Report LSV-98-9, LSV. http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rr-lsv-1998-9.rr.ps.

DERSHOWITZ N. [1990], 'Canonical sets of horn clauses (extended abstract)', University of Illinois.

DERSHOWITZ N. AND JOUANNAUD J.-P. [1990], Rewrite systems, *in* J. van Leeuwen, ed., 'Handbook of Theoretical Computer Science', Vol. B, North-Holland, pp. 243–309.

FRIBOURG L. [1984], A narrowing procedure for theories with constructors, *in* R. Shostak, ed., 'Proc. 7th Int. Conf. on Automated Deduction', Vol. 170 of *Lecture Notes in Computer Science*, Springer-Verlag, Napa, CA., pp. 259–281.

FRIBOURG L. [1986], A strong restriction of the inductive completion procedure, *in* 'Proc. 13th ICALP, Rennes, LNCS 226', Springer-Verlag, pp. 105–115.

FRIBOURG L. [1989], 'A strong restriction of the inductive completion procedure', *Journal of Symbolic Computation* **8**, 253–276.

GANZINGER H. AND STUBER J. [1992], Inductive theorem proving by consistency for first-order clauses, *in* 'Proc. CTRS'92', number 656 *in* 'LNCS', pp. 226–241.

GOGUEN J. A. [1980], How to prove inductive hypothesis without induction, *in* 'Proc. 5th Conf. on Automated Deduction, Les Arcs, France, LNCS 87'.

GRAMLICH B. [1989], Inductive theorem proving using refined unfailing completion techniques, SEKI Report SR-89-14, Fachbereich Informatik, Universität Kaiserslautern.

GRAMLICH B. [1990*a*], Completion based inductive theorem proving: A case study in verifying sorting algorithms, SEKI Report SR-90-04, Fachbereich Informatik, Universität Kaiserslautern.

GRAMLICH B. [1990*b*], Completion based inductive theorem proving: An abstract framework and its applications, *in* L. Aiello, ed., 'Proc. 9th European Conf. on Artificial Intelligence', Pitman Publishing, London, pp. 314–319.

GRAMLICH B. [1990*c*], UNICOM: a refined completion based inductive theorem prover, *in* 'Proc. 10th Int. Conf. on Automated Deduction (CADE 90)', Vol. 449 of *LNAI*, Kaiserslautern, pp. 655–656.

GRAMLICH B. AND LINDNER W. [1991], A guide to UNICOM, an inductive theorem prover based on rewriting and completion techniques, SEKI-Report SR-91-17, Fachbereich Informatik, Universität Kaiserslautern.

GUTTAG J. V. AND HORNING J. J. [1978], 'The algebraic specification of abstract data types', *Acta Informatica* **10**, 27–52.

HUET G. AND HULLOT J.-M. [1982], 'Proofs by induction in equational theories with constructors', *Journal of Computer and System Sciences* **25**(2).

JOUANNAUD J.-P. AND BOUHOULA A. [1997], Automata-driven automated induction, *in* 'Twelfth Annual IEEE Symposium on Logic in Computer Science', IEEE Comp. Soc. Press, Warsaw,Poland.

JOUANNAUD J.-P. AND KOUNALIS E. [1986], Automatic proofs by induction in equational theories without constructors, *in* 'Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.'.

JOUANNAUD J.-P. AND KOUNALIS E. [1989], 'Automatic proofs by induction in theories without constructors', *Information and Computation* **82**(1).

KAPUR D. AND MUSSER D. [1987], 'Proof by consistency', *Artificial Intelligence* **31**(2).

KAPUR D., NARENDRAN P., ROSENKRANTZ D. AND ZHANG H. [1991], 'Sufficient completeness, ground reducibility and their complexity', *Acta Informatica* **28**, 311–350.

KAPUR D., NARENDRAN P. AND ZHANG H. [1987], 'On sufficient completeness and related properties of term rewriting systems', *Acta Informatica* **24**(4), 395–415.

KAPUR D. AND ZHANG H. [1995], 'An overview of the rewrite rule laboratory', *Journal of Mathematics of Computation* .

KAPUR D. AND ZHANG H. [n.d.], Automating induction: Explicit vs inductionless. Unpublished draft.

KÜCHLIN W. [1987], Inductive completion by ground proof transformation, *in* H. Ait-Kaci and M. Nivat, eds, 'Proc. Coll. on the Resolution of Equations in Algebraic Structures, Lakeway', Academic Press.

KOUNALIS E. [1992], 'Testing for the ground (co)-reducibility in term rewriting systems', *Theoretical Computer Science* **106**(1), 87–117.

LANKFORD D. S. [1981], A simple explanation of inductionless induction, Technical Report MTP-14, Mathematics Department, Louisiana Tech. Univ.

MUSSER D. [1980], Proving inductive properties of abstract data types, *in* 'Proc. 7th ACM Symp. on Principles of Programming Languages, Las Vegas'.

NIEUWENHUIS R. AND RUBIO A. [1995], 'Theorem proving with ordering and equality constrained clauses', *Journal of Symbolic Computation* **19**(4), 321–351.

PLAISTED D. [1985], 'Semantic confluence tests and completion methods', *Information and Control* **65**, 182–215.

REDDY U. [1990], Term rewriting induction, *in* 'Proc. 10th Int. Conf. on Automated Deduction, Kaiserslautern, LNCS 449'.

SLAGLE J. R. [1974], 'Automated theorem-proving for theories with simplifiers, commutativity, and associativity', *J. of the Association for Computing Machinery* **21**(4), 622–642.

WALDMANN U. [1992], 'Semantics of order-sorted specifications', *Theoretical Computer Science* **94**, 1–35.

ZHANG H. [1988], Reduction, Superposition and Induction: Automated Reasoning in Equational Logic, PhD thesis, Rensselaer Polytechnic Institute, Troy, N.Y.

Index

Topic index